深入阐释Android开发精髓，真实再现项目开发实况



王翠萍◎编著

Android 经典项目开发实战

- 18个经典项目，涵盖Android应用开发的主流领域。
- 实录560分钟、75个项目开发高清学习视频。
- 完整再现一个个经典项目的开发全程，快速提升实战水平。
- 教授精髓，精讲精炼。赠送源码，拿来就用。



- 15个Android综合项目开发案例
- 38个Android应用开发学习视频

清华大学出版社

Android 经典项目开发实战

王翠萍　编著

清华大学出版社

北　京

内 容 简 介

Android 系统从诞生到现在，短短几年时间，便凭借其操作易用性和开发的简洁性，赢得了广大用户和开发者的支持。截至 2014 年 9 月 30 日，Android 系统的市场占有率高达 85%。

本书共计 18 章，循序渐进地讲解开发 Android 经典项目的具体过程。本书从蓝牙通信系统开始，依次讲解了移动微信系统、移动邮件系统、移动微博系统、网络 RSS 阅读器、开发一个音乐播放器、魔塔游戏、NBA 激情投篮、象棋游戏、暴走轨迹计步器、智能楼宇灯光控制系统、网络防火墙系统、Map 地图、QQ 聊天记录查看器、吃货选择器、智能心率计、仿陌陌交友系统及开发一个 Android 优化系统的具体实现流程，彻底剖析了一个个经典项目的完整实现过程。本书几乎涵盖了所有领域的 Android 项目，讲解方法通俗易懂并且详细，不但适合高手学习，也特别有利于初学者学习并消化。

本书适合 Android 学习者、Android 硬件开发者、Android 物联网开发人员、Android 爱好者、Android 应用开发人员学习，也可以作为相关培训学校和大专院校相关专业的教学用书。

图书在版编目（CIP）数据

Android 经典项目开发实战/王翠萍编著. —北京：清华大学出版社，2015
ISBN 978-7-302-40130-8

I. ①A…　II. ①王…　III. ①移动终端-应用程序-程序设计　IV. ①TN929.53

中国版本图书馆 CIP 数据核字（2015）第 089672 号

责任编辑： 朱英彪
封面设计： 刘　超
版式设计： 刘艳庆
责任校对： 王　颖
责任印制：

出版发行： 清华大学出版社
网　　址： http://www.tup.com.cn，http://www.wqbook.com
地　　址： 北京清华大学学研大厦 A 座　　**邮　　编：** 100084
社 总 机： 010-62770175　　**邮　　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销： 全国新华书店
开　　本： 203mm×260mm　**印　　张：** 39.5　**字　　数：** 1217 千字
（附 DVD 光盘 1 张）
版　　次： 2015 年 7 月第 1 版　**印　　次：** 2015 年 7 月第 1 次印刷
印　　数： 1～3000
定　　价： 88.00 元

产品编号：061880-01

前　　言

2007 年 11 月 5 日，谷歌公司宣布基于 Linux 平台的开源手机操作系统 Android 诞生，该平台号称是首个为移动终端打造的真正开放和完整的移动软件，本书作者将和广大读者一起共同领略这款系统的神奇之处。

市场占有率高居第一

截至 2014 年 9 月，Android 在手机市场上的占有率从 2013 年的 68.8%上升到 85%。而 iOS 则从去年的 19.4%下降到 15.5%，WP 系统从原来的 2.7%，小幅上升至 3.6%。从数据上看，Android 平台占据了市场的主导地位。

由数据可以看出 Android 市场的占有率增加幅度较大，WP 市场小幅增长，但 iOS 却有所下降。就目前来看，智能手机的市场已经饱和，大多数用户都在各个平台中转换。而就在这样一个市场上，Android 还增长了 10%左右的占有率确实不易。

为开发人员提供了平台

（1）保证开发人员可以迅速转型进行 Android 应用开发

Android 应用程序是通过 Java 语言开发的，开发人员只要具备 Java 开发基础，就能很快上手并掌握。作为单独的 Android 应用开发，对 Java 编程门槛的要求并不高，即使是没有编程经验的“门外汉”，也可以在突击学习 Java 之后学习 Android。另外，Android 完全支持 2D、3D 和数据库，并且和浏览器实现了集成。所以通过 Android 平台，程序员可以迅速、高效地开发出绚丽多彩的应用，例如，常见的工具、浏览器和游戏等。

（2）定期举办奖金丰厚的 Android 大赛

为了吸引更多的用户使用 Android 开发程序，谷歌已经成功举办了奖金为数千万美元的开发者竞赛，鼓励开发人员创建出创意十足、十分有用的软件。这种大赛对开发人员来说，不但能练习自己的开发技术，并且高额的奖金也是学员们学习的动力。

（3）开发人员可以利用自己的作品赚钱

为了能让 Android 平台吸引更多的关注，谷歌提供了一个专门下载 Android 应用的门店 Android Market，网址是 https://play.google.com/store。在这个门店中允许开发人员发布应用程序，也允许 Android 用户下载自己喜欢的程序。作为开发者，需要申请开发者账号，申请后才能将自己的程序上传到 Android Market，并且可以对自己的软件进行定价。只要所开发的软件程序足够吸引人，就可以获得很可观的金钱回报。这样实现了学习和赚钱两不误，吸引了更多开发人员加入到 Android 大军中来。

本书的内容

本书共 18 章，循序渐进地讲解了开发 Android 经典项目的具体过程。本书从蓝牙通信系统开始，依次讲解了移动微信系统、移动邮件系统、移动微博系统、网络 RSS 阅读器、开发一个音乐播放器、魔塔游戏、NBA 激情投篮、象棋游戏、暴走轨迹计步器、智能楼宇灯光控制系统、网络防火墙系统、Map 地图、QQ 聊天记录查看器、吃货选择器、智能心率计、仿陌陌交友系统及开发一个 Android 优化系统的具体实现流程，彻底剖析了一个个经典项目的完整实现过程。本书几乎涵盖了所有领域的 Android 项目，讲解通俗易懂并且详细，不但适合高手学习，也特别有利于初学者学习和消化。

本书的版本

Android 系统自 2008 年 9 月发布第一个版本 1.1，截至 2014 年 10 月发布最新版本 5.0，一共有十多个版本。由此可见，Android 系统升级频率较快，一年之中最少有两个新版本诞生。如果过于追求新版本，将力不从心。所以建议广大读者不必追求最新的版本，只需关注最流行的版本即可。据官方统计，截至 2014 年 10 月 25 日，占据前 3 位的版本分别是 Android 4.3、Android 4.4 和 Android 4.2，其实这 3 个版本的区别并不是很大，只是在某领域的细节上进行了更新。为了及时体验 Android 系统的最新功能，本书使用的版本是主流的 Android 5.0。

本书特色

本书内容十分丰富，我们的目标是通过一本图书，提供多本图书的价值，读者可以根据自己的需要有选择地阅读。在内容的编写上，本书具有以下特色。

（1）内容全面，讲解细致

本书几乎涵盖了开发 Android 项目所需要的所有主要知识点，详细讲解了每一个经典项目的实现过程和具体移植方法。每一个知识点都力求用详实和易懂的语言展现在读者面前。

（2）遵循合理的主线进行讲解

为了使读者彻底弄清楚 Android 项目开发的各个知识点，在讲解每一个实例时，都先从项目介绍和规划讲起，然后实现具体编码工作，一直到最终的项目测试。在整个讲解过程讲解了项目的开发技巧和注意事项，实现了综合项目开发大揭秘的目标。

（3）章节独立，自由阅读

本书每一章内容都可以独立成书，读者既可以按照本书编排的章节顺序进行学习，也可以根据自己的需求对某一章节针对性地学习，相信阅读本书会带来很大的快乐。

（4）实例典型，实用性强

本书讲解了现实中最典型 Android 项目的实现方法和架构技巧，这些应用都是在商业项目中最需要的部分。读者可以直接将本书中的知识应用到自己的项目中，实现无缝对接。

（5）超值学习光盘

本书光盘中，除配备了 18 个经典项目详细的源代码、教学 PPT 之外，还实录了 75 个高清学习视频，既有相关知识点讲解视频，也有详细的项目开发过程。除此以外，本书额外赠送了 38 个 Android



应用开发学习视频，以及 15 个 Android 应用开发综合案例，包括仿小米录音机、音乐播放器、跟踪定位系统、仿陌陌交友系统、手势音乐播放器、智能家居系统、湿度测试仪、象棋游戏、抢滩登陆游戏、九宫格数独游戏、健康饮食系统、仓库管理系统、个人财务系统、仿去哪儿酒店预定系统、仿开心网客户端等。通过这些附配资源，读者的学习过程会更加方便、快捷。

（6）配备 Eclipse 和 Android Studio 双环境源码

自 Android 6.0 以来，谷歌公司不再对 Eclipse 环境提供技术支持，而是主推 Android Studio 环境，因此本书为广大读者提供了详细的 Android Studio 学习教程，以及 Eclispe、Android Studio 双重环境下的源码，读者可登录本书的服务论坛 http://www.chubanbook.com 进行下载。

读者对象

本书适合 Android 学习者、Android 硬件开发者、Android 物联网开发人员、Android 爱好者、Android 应用开发人员学习，也可以作为相关培训学校和大专院校相关专业的教学用书。

参与本书编写的人员还有周秀、付松柏、邓才兵、钟世礼、谭贞军、张加春、王教明、万春潮、郭慧玲、侯恩静、程娟、王文忠、陈强、何子夜、李天祥、周锐、朱桂英、张元亮、张韶青、秦丹枫。本团队在编写过程中，得到了清华大学出版社工作人员的大力支持，正是各位编辑的求实、耐心和效率，才使本书在这么短的时间内出版。另外，十分感谢我的家人在我写作时给予的巨大支持。由于水平有限，纰漏和不尽如人意之处在所难免，诚请读者提出意见或建议，以便修订使之更臻完善。我们提供了售后支持网站（http://www.chubanbook.com）及 QQ 群（192153124），读者朋友如有疑问可以在此提出，一定会得到满意的答复。

编　者

目　录





第 1 章　蓝牙通信系统

蓝牙（Bluetooth）是一种支持设备短距离（一般 10m 内）通信的无线电技术，能在移动电话、PDA、无线耳机、笔记本电脑、相关外设等众多设备之间进行信息无线交换。本章将首先讲解 Android 系统中蓝牙模块的底层源码和实现原理，然后详细讲解在 Android 平台中开发一个蓝牙通信系统的过程，为读者学习本书后面的知识打下基础。

1.1　蓝 牙 介 绍

知识点讲解：光盘:视频\视频讲解\第 1 章\蓝牙介绍.avi

“蓝牙”系统比较复杂，但是又比较常用，要想完全掌握蓝牙应用开发技术，需要先了解其底层结构。在本节的内容中，将简要讲解蓝牙系统底层结构的基本知识。

1.1.1　蓝牙概述

利用“蓝牙”技术，能够有效地简化移动通信终端设备之间的通信，也能够成功地简化设备与 Internet 之间的通信，从而使数据传输变得更加迅速高效。蓝牙采用分散式网络结构以及快跳频和短包技术，支持点对点及点对多点通信，工作在全球通用的 2.4GHz ISM（即工业、科学、医学）频段。其数据传输速率为 1Mbps，采用时分双工传输方案实现全双工传输。

1. 发展历程

“蓝牙”这一名称来自于十世纪的一位丹麦国王 Harald Blatand，Blatand 在英文里的意思可以被解释为 Bluetooth。因为国王喜欢吃蓝莓，牙龈每天都是蓝色的，所以叫蓝牙。蓝牙的创始者是瑞典爱立信公司，早在 1994 年就已开始研发。1997 年，爱立信与其他设备生产商联系，并激发了他们对该项技术的浓厚兴趣。1998 年 2 月，5 个跨国大公司，包括爱立信、诺基亚、IBM、东芝及 Intel 组成了一个特殊兴趣小组（SIG），他们共同的目标是建立一个全球性的小范围无线通信技术，即现在的蓝牙。

Bluetooth 无线技术规格供全球的成员公司免费使用。许多行业的制造商都积极地在其产品中实施此技术，以减少使用零乱的电线，来实现无缝连接、传输立体声、传输数据或进行语音通信。Bluetooth 技术在 2.4 GHz 波段运行，该波段是一种无需申请许可证的工业、科技、医学（ISM）无线电波段。正因如此，使用 Bluetooth 技术不需要支付任何费用，但必须向手机提供商注册使用 GSM 或 CDMA，除了设备费用外，用户不需要为使用 Bluetooth 技术再支付任何费用。

Bluetooth 技术得到了空前广泛的应用，集成该技术的产品从手机、汽车到医疗设备，使用该技术的用户从消费者、工业市场到企业等，不一而足。低功耗、小体积以及低成本的芯片解决方案使得 Bluetooth 技术甚至可以应用于极微小的设备中。

2. 特点

Bluetooth 技术是一项即时技术，不要求固定的基础设施，且易于安装和设置。用户不需要电缆即可实

现连接。新用户使用亦不费力，只需拥有 Bluetooth 品牌产品，检查可用的配置文件，将其连接至使用同一配置文件的另一 Bluetooth 设备即可。后续的 PIN 码流程就如同在 ATM 机器上操作一样简单。用户外出时，可以随身带上个人局域网（PAN），甚至可以与其他网络连接。

1.1.2 Android 中的蓝牙系统

Android 包含了对蓝牙网络协议栈的支持，这使得蓝牙设备能够无线连接其他蓝牙设备交换数据。Android 的应用程序框架提供了访问蓝牙功能的 APIs。这些 APIs 让应用程序能够无线连接其他蓝牙设备，实现点对点或点对多点的无线交互功能。通过使用蓝牙 APIs，一个 Android 应用程序能够实现如下功能。

☑ 扫描其他蓝牙设备。
☑ 查询本地蓝牙适配器（local Bluetooth adapter），用于配对蓝牙设备。
☑ 建立 RFCOMM 信道（channels）。
☑ 通过服务发现（service discovery）连接其他设备。
☑ 数据通信。
☑ 管理多个连接。

1.2 Android 蓝牙系统的层次结构

知识点讲解：光盘:视频\视频讲解\第 1 章\Android 蓝牙系统的层次结构.avi

Android 平台的蓝牙系统是基于 BlueZ，通过 Linux 中一套完整的蓝牙协议栈开源实现的。当前 BlueZ 被广泛应用于各种 Linux 版本中，并被芯片公司移植到各种芯片平台上使用。在 Linux 2.6 内核中已经包含了完整的 BlueZ 协议栈，在 Android 系统中已经移植并嵌入了 BlueZ 的用户空间实现，并且随着硬件技术的发展而不断更新。

蓝牙技术实际上是一种短距离无线电技术。在 Android 系统中的蓝牙除了使用 kernel 支持外，还需要用户空间的 BlueZ 的支持。

Android 平台中蓝牙系统的基本层次结构如图 1-1 所示。

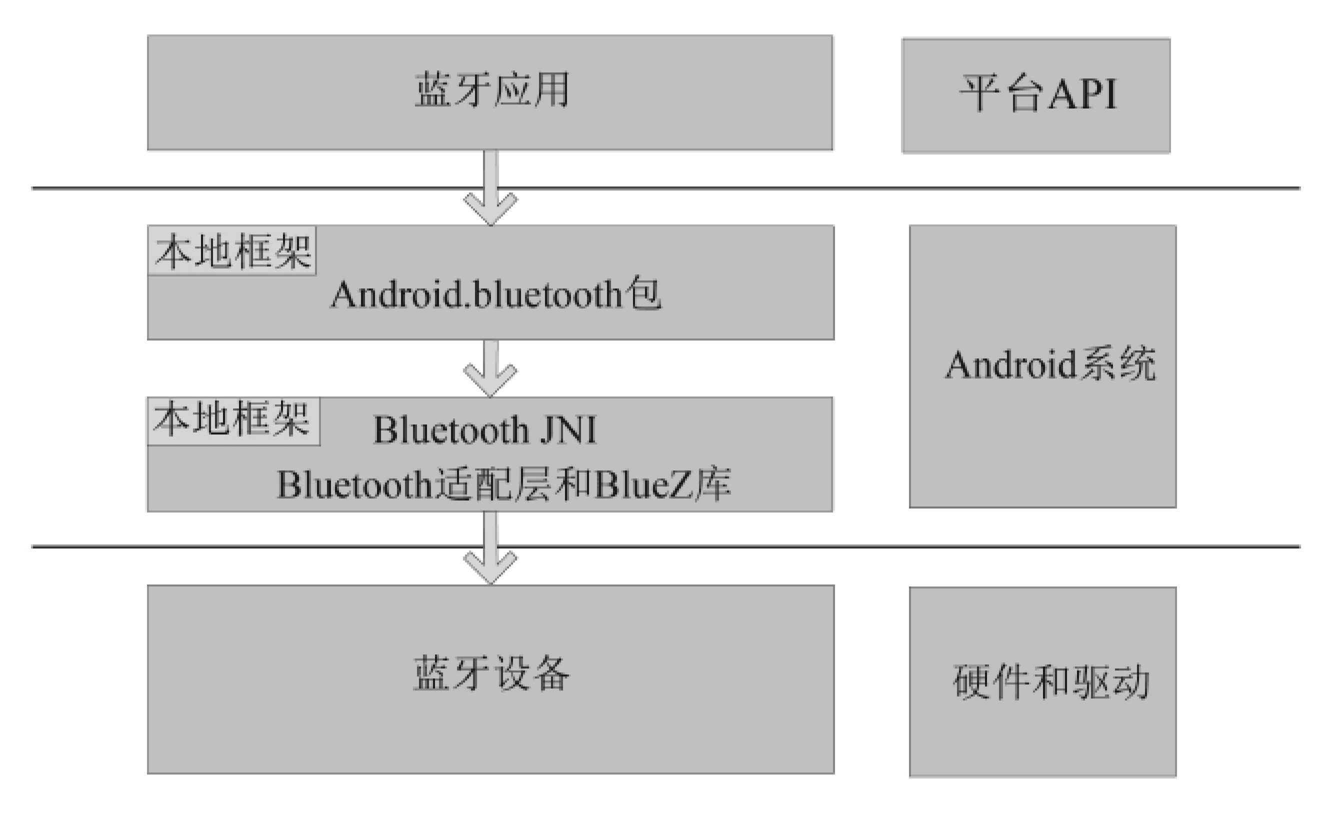


图 1-1　蓝牙系统的层次结构

Android 平台中蓝牙系统从上到下主要包括 Java 框架中的 Bluetooth 类、Android 适配库、BlueZ 库、驱动程序和协议，这几部分的系统结构如图 1-2 所示。

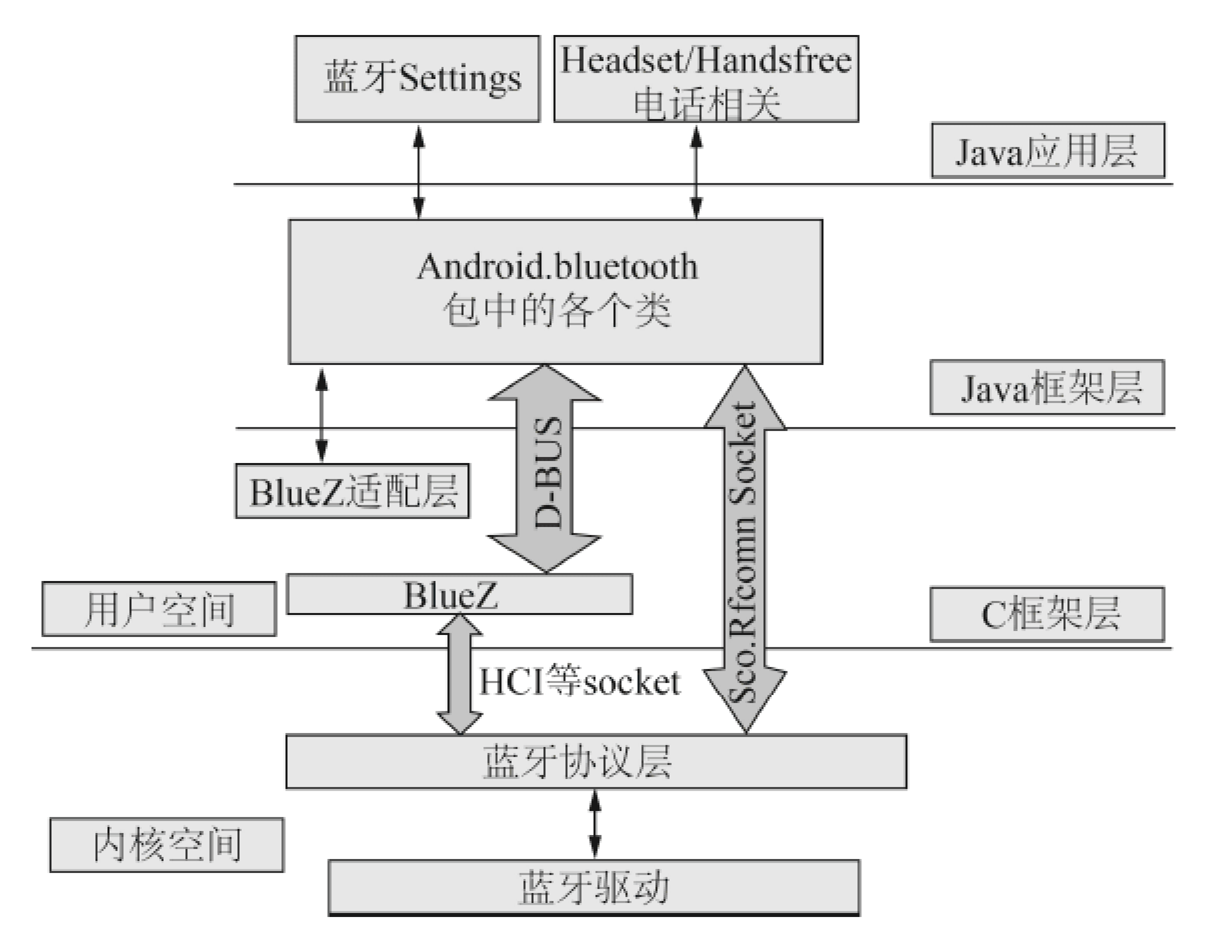


图 1-2　蓝牙系统结构

图 1-2 中各个层次结构的具体说明如下所示。

（1）BlueZ 库

Android 蓝牙设备管理的库的路径如下所示。

```
external/bluez/
```

可以分别生成 libbluetooth.so、libbluedroid.so 和 hcidump 等众多相关工具和库。BlueZ 库提供了对用户空间蓝牙的支持，其中包含了主机控制协议 HCI 以及其他众多内核实现协议的接口，并且实现了所有蓝牙应用模式 Profile。

（2）蓝牙的 JNI 部分

此部分的代码路径如下所示。

```
frameworks/base/core/jni/
```

（3）Java 框架层

Java 框架层的实现代码保存在如下路径。

```
frameworks/base/core/java/android/bluetooth        //蓝牙部分对应应用程序的 API
frameworks/base/core/java/android/Server           //蓝牙的服务部分
```

蓝牙的服务部分负责管理并使用底层本地服务，并封装成系统服务。而在 android.bluetooth 部分中包含了各个蓝牙平台的 API 部分，以供应用程序层使用。

（4）Bluetooth 的适配库

Bluetooth 适配库的代码路径如下所示。

```
system/bluetooth/
```

此层用于生成库 libbluedroid.so 以及相关工具和库，能够实现对蓝牙设备的管理，例如蓝牙设备的电源管理。

在 Android 系统的如下目录中实现了 libbluedroid。

```
system/bluetooth/
```

可以调用 rfkill 接口来控制电源管理，如果已经实现了 rfkill 接口，则无需再进行配置。如果在文件 init.rc 中已经实现了 hciattach 服务，则说明在 libbluedroid 中已经实现对其调用以操作蓝牙的初始化。

1.3 和蓝牙相关的类

知识点讲解：光盘:视频\视频讲解\第 1 章\和蓝牙相关的类.avi

本章前面已经介绍了 Android 系统中蓝牙的基本知识，本节将详细讲解在 Android 系统中和蓝牙相关的类，为读者学习本书后面的知识打好基础。

1.3.1 BluetoothSocket 类

1. BluetoothSocket 类基础

BluetoothSocket 类的格式如下所示。

```
public static class Gallery.LayoutParams extends ViewGroup.LayoutParams
```

BluetoothSocket 类的结构如下所示。

```
java.lang.Object
android.view. ViewGroup.LayoutParams
android.widget.Gallery.LayoutParams
```

Android 的蓝牙系统和 Socket 套接字密切相关，蓝牙端的监听接口和 TCP 的端口类似，都是使用了 Socket 和 ServerSocket 类。在服务器端，使用 BluetoothServerSocket 类来创建一个监听服务端口。当一个连接被 BluetoothServerSocket 所接受，会返回一个新的 BluetoothSocket 来管理该连接。在客户端，使用一个单独的 BluetoothSocket 类去初始化一个外接连接和管理该连接。

最通常使用的蓝牙端口是 RFCOMM，它是被 Android API 支持的类型。RFCOMM 是一个面向连接，通过蓝牙模块进行的数据流传输方式，也被称为串行端口规范（Serial Port Profile，SPP）。

为了创建一个 BluetoothSocket 去连接到一个已知设备，使用 BluetoothDevice.createRfcommSocketToServiceRecord()方法，然后调用 connect()方法尝试一个面向远程设备的连接。这个调用将被阻塞，直到一个连接已经建立或者该连接失效。

创建一个 BluetoothSocket 作为服务端（或者“主机”），每当该端口连接成功后，无论初始化为客户端，或者被接受作为服务器端，都通过方法 getInputStream()和 getOutputStream()来打开 I/O 流，从而获得各自的 InputStream 和 OutputStream 对象。

BluetoothSocket 类的线程是安全的，因为 close()方法总会马上放弃外界操作并关闭服务器端口。

2. BluetoothSocket 类的公共方法

（1）public void close ()

- ☑ 功能：马上关闭该端口并且释放所有相关的资源。在其他线程的该端口中引起阻塞，从而使系统马上抛出一个 I/O 异常。
- ☑ 异常：IOException。

（2）public void connect ()

- ☑ 功能：尝试连接到远程设备。该方法将阻塞，指导一个连接建立或者失效。如果该方法没有返回异常值，则该端口现在已经建立。当设备查找正在进行时，创建对远程蓝牙设备的新连接不可被尝试。设备查找在蓝牙适配器上是一个重量级过程，并且肯定会降低一个设备的连接。使用 cancelDiscovery()方法会取消一个外界的查询，因为这个查询并不由活动管理，而是作为一个系统



服务来运行，所以即使它不能直接请求一个查询，应用程序也总会调用 cancelDiscovery()方法。使用 close()方法可以放弃从另一线程而来的调用。

☑ 异常：IOException，表示一个错误，例如连接失败。

（3）public InputStream getInputStream ()

☑ 功能：通过连接的端口获得输入数据流，即使该端口未连接，该输入数据流也会返回。不过在该数据流上的操作将抛出异常，直到相关的连接已经建立。

☑ 返回值：输入流。

☑ 异常：IOException。

（4）public OutputStream getOutputStream ()

☑ 功能：通过连接的端口获得输出数据流，即使该端口未连接，该输出数据流也会返回。不过在该数据流上的操作将抛出异常，直到相关的连接已经建立。

☑ 返回值：输出流。

☑ 异常：IOException。

（5）public BluetoothDevice getRemoteDevice ()

☑ 功能：获得该端口正在连接或者已经连接的远程设备。

☑ 返回值：远程设备。

1.3.2　BluetoothServerSocket 类

1. BluetoothServerSocket 类基础

BluetoothServerSocket 类的格式如下所示。

```
public final class BluetoothServerSocket extends Object implements Closeable
```

BluetoothServerSocket 类的结构如下所示。

```
java.lang.Object
android.bluetooth.BluetoothServerSocket
```

2. BluetoothServerSocket 类的公共方法

（1）public BluetoothSocketaccept (int timeout)

☑ 功能：阻塞直到超时时间内的连接建立。在一个成功建立的连接上返回一个已连接的 BluetoothSocket 类。每当该调用返回时，可以在此调用接收以后新来的连接。close()方法可以用来放弃从另一线程来的调用。

☑ 参数 timeout：表示阻塞超时时间。

☑ 返回值：已连接的 BluetoothSocket。

☑ 异常：IOException，表示出现错误，例如该调用被放弃或超时。

（2）public BluetoothSocket accept ()

☑ 功能：阻塞直到一个连接已经建立。在一个成功建立的连接上返回一个已连接的 BluetoothSocket 类。每当该调用返回时，可以在此调用接收以后新来的连接。使用 close()方法可以用来放弃从另一线程来的调用。

☑ 返回值：已连接的 BluetoothSocket。

☑ 异常：IOException，表示出现错误，例如该调用被放弃或者超时。

（3）public void close ()

☑ 功能：马上关闭端口，并释放所有相关的资源。在其他线程的该端口中引起阻塞，从而使系统马上

抛出一个 I/O 异常。关闭 BluetoothServerSocket 不会关闭接受自 accept()的任意 BluetoothSocket。

☑ 异常：IOException。

1.3.3 BluetoothAdapter 类

1. BluetoothAdapter 类基础

BluetoothAdapter 类的格式如下所示。

```
public final class BluetoothAdapter extends Object
```

BluetoothAdapter 类的结构如下所示。

```
java.lang.Object
android.bluetooth.BluetoothAdapter
```

BluetoothAdapter 代表本地的蓝牙适配器设备，通过此类用户能够执行基本的蓝牙操作。例如，初始化设备的搜索，查询可匹配的设备集，使用一个已知的 MAC 地址来初始化一个 BluetoothDevice 类，创建一个 BluetoothServerSocket 类以监听其他设备对本机的连接请求等。

为了得到这个代表本地蓝牙适配器的 BluetoothAdapter 类，需要调用静态方法 getDefaultAdapter()，这是所有蓝牙动作使用的第一步。当拥有本地适配器以后，用户可以获得一系列的 BluetoothDevice 对象，这些对象代表所有拥有 getBondedDevice()方法的已经匹配的设备；用 startDiscovery()方法来开始设备的搜寻；或者创建一个 BluetoothServerSocket 类，通过 listenUsingRfcommWithServiceRecord(String, UUID)方法来监听新来的连接请求。

注意： 大部分方法需要BLUETOOTH权限，一些方法同时需要BLUETOOTH_ADMIN权限。

2. BluetoothAdapter 类的常量

（1）String ACTION_DISCOVERY_FINISHED

☑ 广播事件：本地蓝牙适配器已经完成设备的搜寻过程，需要 BLUETOOTH 权限接收。

☑ 常量值：android.bluetooth.adapter.action.DISCOVERY_FINISHED。

（2）String ACTION_DISCOVERY_STARTED

☑ 广播事件：本地蓝牙适配器已经开始对远程设备的搜寻过程，通常牵涉到一个大概需时 12 秒的查询扫描过程，紧跟着是一个对每个获取到自身蓝牙名称的新设备的页面扫描。用户会发现一个把 ACTION_FOUND 常量通知为远程蓝牙设备的注册。设备查找是一个重量级过程，当查找正在进行时，用户不能尝试对新的远程蓝牙设备进行连接，同时存在的连接将获得有限制的带宽以及高等待时间。用户可用 cancelDiscovery()类来取消正在执行的查找进程。需要 BLUETOOTH 权限接收。

☑ 常量值：android.bluetooth.adapter.action.DISCOVERY_STARTED。

（3）String ACTION_LOCAL_NAME_CHANGED

☑ 广播活动：本地蓝牙适配器已经更改了其蓝牙名称，该名称对远程蓝牙设备是可见的，总是包含一个带有名称的 EXTRA_LOCAL_NAME 附加域，需要 BLUETOOTH 权限接收。

☑ 常量值：android.bluetooth.adapter.action.LOCAL_NAME_CHANGED。

（4）String ACTION_REQUEST_DISCOVERABLE

☑ Activity 活动：显示一个请求被搜寻模式的系统活动。如果蓝牙模块当前未打开，该活动也将请求用户打开蓝牙模块。被搜寻模式和 SCAN_MODE_CONNECTABLE_DISCOVERABLE 等价。当远程设备执行查找进程时，允许其发现该蓝牙适配器。从隐私安全考虑，Android 不会将被搜寻模式设置为默认状态。该意图的发送者可以选择性地运用 EXTRA_DISCOVERABLE_DURATION 这个

附加域去请求发现设备的持续时间。普遍来说，对于每一请求，默认的持续时间为 120 秒，最大值则可达到 300 秒。

Android 运用 onActivityResult(int, int, Intent)回收方法来传递该活动结果的通知。被搜寻的时间（以秒为单位）将通过 resultCode 值来显示，如果用户拒绝被搜寻，或者设备产生了错误，则通过 RESULT_CANCELED 值来显示。

每当扫描模式变化时，应用程序可以通过 ACTION_SCAN_MODE_CHANGED 值来监听全局的消息通知。例如，当设备停止被搜寻以后，该消息可以被系统通知给应用程序，需要 BLUETOOTH 权限。

☑ 常量值：android.bluetooth.adapter.action.REQUEST_DISCOVERABLE。

（5）String ACTION_REQUEST_ENABLE

☑ Activity 活动：显示一个允许用户打开蓝牙模块的系统活动。当蓝牙模块完成打开工作，或者当用户决定不打开蓝牙模块时，系统活动将返回该值。Android 运用 onActivityResult(int, int, Intent)回收方法来传递该活动结果的通知。如果蓝牙模块被打开，将通过 resultCode 值 RESULT_OK 来显示；如果用户拒绝该请求，或者设备产生了错误，则通过 RESULT_CANCELED 值来显示。每当蓝牙模块被打开或者关闭，应用程序可以通过 ACTION_STATE_CHANGED 值来监听全局的消息通知，需要 BLUETOOTH 权限。

☑ 常量值：android.bluetooth.adapter.action.REQUEST_ENABLE。

（6）String ACTION_SCAN_MODE_CHANGED

☑ 广播活动：指明蓝牙扫描模块或者本地适配器已经发生变化。总是包含 EXTRA_SCAN_MODE 和 EXTRA_PREVIOUS_SCAN_MODE，这两个附加域各自包含了新的和旧的扫描模式，需要 BLUETOOTH 权限。

☑ 常量值：android.bluetooth.adapter.action.SCAN_MODE_CHANGED。

（7）String ACTION_STATE_CHANGED

☑ 广播活动：原来的蓝牙适配器的状态已经改变，例如，蓝牙模块已经被打开或者关闭。它总是包含 EXTRA_STATE 和 EXTRA_PREVIOUS_STATE，这两个附加域各自包含了新的和旧的状态，需要 BLUETOOTH 权限接收。

☑ 常量值：android.bluetooth.adapter.action.STATE_CHANGED。

（8）int ERROR

☑ 功能：标记该类的错误值。确保和该类中的任意其他整数常量不相等。为需要一个标记错误值的函数提供了便利。例如：

```
Intent.getIntExtra(BluetoothAdapter.EXTRA_STATE, BluetoothAdapter.ERROR)
```

☑ 常量值：-2147483648 (0x80000000)。

（9）String EXTRA_DISCOVERABLE_DURATION

☑ 功能：试图在 ACTION_REQUEST_DISCOVERABLE 常量中作为一个可选的整型附加域，来为短时间内的设备发现请求一个特定的持续时间。默认值为 120 秒，超过 300 秒的请求将被限制。这些值是可以变化的。

☑ 常量值：android.bluetooth.adapter.extra.DISCOVERABLE_DURATION。

（10）String EXTRA_LOCAL_NAME

☑ 功能：试图在 ACTION_LOCAL_NAME_CHANGED 常量中作为一个字符串附加域，来请求本地蓝牙的名称。

☑ 常量值：android.bluetooth.adapter.extra.LOCAL_NAME。

（11）String EXTRA_PREVIOUS_SCAN_MODE

☑ 功能：试图在 ACTION_SCAN_MODE_CHANGED 常量中作为一个整型附加域来请求以前的扫描

模式，可以取得值如下所示。

- ➢ SCAN_MODE_NONE
- ➢ SCAN_MODE_CONNECTABLE
- ➢ SCAN_MODE_CONNECTABLE_DISCOVERABLE

☑ 常量值：android.bluetooth.adapter.extra.PREVIOUS_SCAN_MODE。

（12）String EXTRA_PREVIOUS_STATE

☑ 功能：试图在 ACTION_STATE_CHANGED 常量中作为一个整型附加域，来请求以前的供电状态，可以取得值如下所示。

- ➢ STATE_OFF
- ➢ STATE_TURNING_ON
- ➢ STATE_ON
- ➢ STATE_TURNING_OFF

☑ 常量值：android.bluetooth.adapter.extra.PREVIOUS_STATE。

（13）String EXTRA_SCAN_MODE

☑ 功能：试图在 ACTION_SCAN_MODE_CHANGED 常量中作为一个整型附加域，来请求当前的扫描模式，可以取得值如下所示。

- ➢ SCAN_MODE_NONE
- ➢ SCAN_MODE_CONNECTABLE
- ➢ SCAN_MODE_CONNECTABLE_DISCOVERABLE

☑ 常量值：android.bluetooth.adapter.extra.SCAN_MODE。

（14）String EXTRA_STATE

☑ 功能：试图在 ACTION_STATE_CHANGED 常量中作为一个整型附加域，来请求当前的供电状态，可以取得值如下所示。

- ➢ STATE_OFF
- ➢ STATE_TURNING_ON
- ➢ STATE_ON
- ➢ STATE_TURNING_OFF

☑ 常量值：android.bluetooth.adapter.extra.STATE。

（15）int SCAN_MODE_CONNECTABLE

☑ 功能：指明在本地蓝牙适配器中，查询扫描功能失效，但页面扫描功能有效。因此该设备不能被远程蓝牙设备发现，但如果以前曾经发现过该设备，则远程设备可以对其进行连接。

☑ 常量值：21（0x00000015）。

（16）int SCAN_MODE_CONNECTABLE_DISCOVERABLE

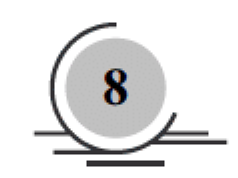
☑ 功能：指明在本地蓝牙适配器中，查询扫描功能和页面扫描功能都有效。因此该设备既可以被远程蓝牙设备发现，也可以被其连接。

☑ 常量值：23（0x00000017）。

（17）int SCAN_MODE_NONE

☑ 功能：指明在本地蓝牙适配器中，查询扫描功能和页面扫描功能都失效。因此该设备既不可以被远程蓝牙设备发现，也不可以被其连接。

☑ 常量值：20（0x00000014）。



（18）int STATE_OFF

☑ 功能：指明本地蓝牙适配器模块已经关闭。

☑ 常量值：10（0x0000000a）。

（19）int STATE_ON

☑ 功能：指明本地蓝牙适配器模块已经打开，并且准备被使用。

（20）int STATE_TURNING_OFF

☑ 功能：指明本地蓝牙适配器模块正在关闭。本地客户端可以立刻尝试友好地断开任意外部连接。

☑ 常量值：13（0x0000000d）。

（21）int STATE_TURNING_ON

☑ 功能：指明本地蓝牙适配器模块正在打开，然而本地客户在尝试使用这个适配器之前需要为 STATE_ON 状态而等待。

☑ 常量值：11（0x0000000b）。

3. BluetoothAdapter 类的公共方法

（1）public boolean cancelDiscovery ()

☑ 功能：取消当前的设备发现查找进程，需要 BLUETOOTH_ADMIN 权限。因为对蓝牙适配器而言，查找是一个重量级的过程，因此这个方法必须在尝试连接到远程设备前使用 connect()方法进行调用。发现的过程不会由活动来进行管理，但是会作为一个系统服务来运行，因此即使不能直接请求这样的一个查询动作，也必须取消该搜索进程。如果蓝牙状态不是 STATE_ON，这个 API 将返回 false。蓝牙打开后，等待 ACTION_STATE_CHANGED 更新成 STATE_ON。

☑ 返回值：成功则返回 true，有错误则返回 false。

（2）public static boolean checkBluetoothAddress (String address)

☑ 功能：验证如 00:43:A8:23:10:F0 之类的蓝牙地址，字母必须为大写才有效。

☑ 参数 address：字符串形式的蓝牙模块地址。

☑ 返回值：地址正确则返回 true，否则返回 false。

（3）public boolean disable ()

☑ 功能：关闭本地蓝牙适配器——不能在没有明确关闭蓝牙的用户动作中使用。该方法将友好地停止所有的蓝牙连接，停止蓝牙系统服务，以及对所有基础蓝牙硬件进行断电。没有用户的直接同意，蓝牙永远不能被禁止。disable()方法只提供了一个应用，该应用包含了一个改变系统设置的用户界面（例如“电源控制”应用）。

☑ 这是一个异步调用方法：该方法将马上获得返回值，用户要通过监听 ACTION_STATE_CHANGED 值来获取随后的适配器状态改变的通知。如果该调用返回 true 值，则该适配器状态会立刻从 STATE_ON 转向 STATE_TURNING_OFF，稍后则会转为 STATE_OFF 或者 STATE_ON。如果该调用返回 false，那么系统已经有一个保护蓝牙适配器被关闭的问题，例如该适配器已经被关闭了。

☑ 需要 BLUETOOTH_ADMIN 权限。

☑ 返回值：如果蓝牙适配器的停止进程已经开启则返回 true，如果产生错误则返回 false。

（4）public boolean enable ()

☑ 功能：打开本地蓝牙适配器——不能在没有明确打开蓝牙的用户动作中使用。该方法将为基础的蓝牙硬件供电，并且启动所有的蓝牙系统服务。没有用户的直接同意，蓝牙永远不能被禁止。如果用户为了创建无线连接而打开了蓝牙模块，则需要 ACTION_REQUEST_ENABLE 值，该值将提出一个请求用户允许以打开蓝牙模块的会话。enable()值只提供了一个应用，该应用包含了一个改变系统设置的用户界面（例如“电源控制”应用）。

☑ 这是一个异步调用方法：该方法将马上获得返回值，用户要通过监听 ACTION_STATE_CHANGED 值来获取随后的适配器状态改变的通知。如果该调用返回 true 值，则该适配器状态会立刻从 STATE_OFF 转向 STATE_TURNING_ON，稍后则会转为 STATE_OFF 或者 STATE_ON。如果该调用返回 false，那么说明系统已经有一个保护蓝牙适配器被打开的问题，例如飞行模式，或者该适配器已经被打开。

☑ 需要 BLUETOOTH_ADMIN 权限。

☑ 返回值：如果蓝牙适配器的打开进程已经开启则返回 true，如果产生错误则返回 false。

（5）public String getAddress ()

☑ 功能：返回本地蓝牙适配器的硬件地址，例如：

```
00:11:22:AA:BB:CC
```

☑ 需要 BLUETOOTH 权限。

☑ 返回值：字符串形式的蓝牙模块地址。

（6）public Set<BluetoothDevice> getBondedDevices ()

☑ 功能：返回已经匹配到本地适配器的 BluetoothDevice 类的对象集合。如果蓝牙状态不是 STATE_ON，这个 API 将返回 false。蓝牙打开后，等待 ACTION_STATE_CHANGED 更新成 STATE_ON。需要 BLUETOOTH 权限。

☑ 返回值：未被修改的 BluetoothDevice 类的对象集合，如果有错误则返回 null。

（7）public static synchronized BluetoothAdapter getDefaultAdapter ()

☑ 功能：获取对默认本地蓝牙适配器的操作权限。目前 Android 只支持一个蓝牙适配器，但是 API 可以被扩展为支持多个适配器。该方法总是返回默认的适配器。

☑ 返回值：返回默认的本地适配器，如果蓝牙适配器在该硬件平台上不能被支持，则返回 null。

（8）public String getName ()

☑ 功能：获取本地蓝牙适配器的蓝牙名称，这个名称对于外界蓝牙设备而言是可见的。需要 BLUETOOTH 权限。

☑ 返回值：该蓝牙适配器名称，如果有错误则返回 null。

（9）public BluetoothDevice getRemoteDevice (String address)

☑ 功能：为给予的蓝牙硬件地址获取一个 BluetoothDevice 对象。合法的蓝牙硬件地址必须为大写，格式类似于 00:11:22:33:AA:BB。checkBluetoothAddress(String)方法可以用来验证蓝牙地址的正确性。BluetoothDevice 类对于合法的硬件地址总会产生返回值，即使这个适配器从未见过该设备。

☑ 参数：address，合法的蓝牙 MAC 地址。

☑ 异常：IllegalArgumentException，如果地址不合法。

（10）public int getScanMode ()

☑ 功能：获取本地蓝牙适配器的当前蓝牙扫描模式，蓝牙扫描模式决定本地适配器可连接并且可被远程蓝牙设备所连接。需要 BLUETOOTH 权限，可能的取值如下所示。

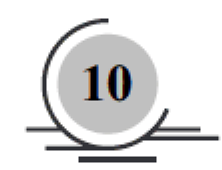
  - SCAN_MODE_NONE
  - SCAN_MODE_CONNECTABLE
  - SCAN_MODE_CONNECTABLE_DISCOVERABLE

如果蓝牙状态不是 STATE_ON，则这个 API 将返回 false。蓝牙打开后，等待 ACTION_STATE_CHANGED 更新成 STATE_ON。

☑ 返回值：扫描模式。

（11）public int getState ()

☑ 功能：获取本地蓝牙适配器的当前状态，需要 BLUETOOTH 类，可能的取值如下所示。



- ➢ STATE_OFF
- ➢ STATE_TURNING_ON
- ➢ STATE_ON
- ➢ STATE_TURNING_OFF

☑ 返回值：蓝牙适配器的当前状态。

（12）public boolean isDiscovering ()

☑ 功能：如果当前蓝牙适配器正处于设备发现查找进程中，则返回真值。设备查找是一个重量级过程。当查找正在进行时，用户不能尝试对新的远程蓝牙设备进行连接，同时存在的连接将获得有限制的带宽以及高等待时间。用户可用 cencelDiscovery()类来取消正在执行的查找进程。

应用程序也可以为 ACTION_DISCOVERY_STARTED 或者 ACTION_DISCOVERY_FINISHED 进行注册，从而当查找开始或者完成时，可以获得通知。

如果蓝牙状态不是 STATE_ON，这个 API 将返回 false。蓝牙打开后，等待 ACTION_STATE_CHANGED 更新成 STATE_ON，需要 BLUETOOTH 权限。

☑ 返回值：如果正在查找，则返回 true。

（13）public boolean isEnabled ()

☑ 功能：如果蓝牙正处于打开状态并可用，则返回真值，与 getBluetoothState()==STATE_ON 等价，需要 BLUETOOTH 权限。

☑ 返回值：如果本地适配器已经打开，则返回 true。

（14）public BluetoothServerSocket listenUsingRfcommWithServiceRecord (String name, UUID uuid)

☑ 功能：创建一个正在监听的安全的带有服务记录的无线射频通信（RFCOMM）蓝牙端口。一个对该端口进行连接的远程设备将被认证，对该端口的通信将被加密。使用 accept()方法可以获取从监听 BluetoothServerSocket 处新来的连接。该系统分配一个未被使用的无线射频通信通道来进行监听。

该系统也将注册一个服务探索协议（SDP）记录，该记录带有一个包含了特定的通用唯一识别码（Universally Unique Identifier，UUID），服务器名称和自动分配通道的本地 SDP 服务。远程蓝牙设备可以用相同的 UUID 来查询自己的 SDP 服务器，并搜寻连接到了哪个通道上。如果该端口已经关闭，或者如果该应用程序异常退出，则这个 SDP 记录会被移除。使用 createRfcommSocketToServiceRecord(UUID)可以从另一使用相同 UUID 的设备来连接到这个端口，需要 BLUETOOTH 权限。

☑ 参数：

- ➢ name：SDP 记录下的服务器名。
- ➢ uuid：SDP 记录下的 UUID。

☑ 返回值：一个正在监听的无线射频通信蓝牙服务端口。

☑ 异常：IOException，表示产生错误，例如蓝牙设备不可用，或者许可无效，或者通道被占用。

（15）public boolean setName (String name)

☑ 功能：设置蓝牙或者本地蓝牙适配器的昵称，这个名字对于外界蓝牙设备而言是可见的。合法的蓝牙名称最多拥有 248 位 UTF-8 字符，但是很多外界设备只能显示前 40 个字符，有些可能只显示前 20 个字符。

如果蓝牙状态不是 STATE_ON，这个 API 将返回 false。蓝牙打开后，等待 ACTION_STATE_CHANGED 更新成 STATE_ON，需要 BLUETOOTH_ADMIN 权限。

☑ 参数 name：一个合法的蓝牙名称。

☑ 返回值：如果该名称已被设定，则返回 true，否则返回 false。

（16）public boolean startDiscovery ()

☑ 功能：开始对远程设备进行查找的进程，通常牵涉到一个大概需时 12 秒的查询扫描过程，紧跟着是一个对每个获取到自身蓝牙名称的新设备的页面扫描。

这是一个异步调用方法，该方法将马上获得返回值，注册 ACTION_DISCOVERY_STARTED 和 ACTION_DISCOVERY_FINISHED 意图准确地确定该探索是处于开始阶段或者完成阶段。注册 ACTION_FOUND 以获得远程蓝牙设备已找到的通知。

设备查找是一个重量级过程。当查找正在进行时，用户不能尝试对新的远程蓝牙设备进行连接，同时存在的连接将获得有限制的带宽以及高等待时间。用户可用 cencelDiscovery()类来取消正在执行的查找进程。发现的过程不会由活动来进行管理，但是会作为一个系统服务来运行，因此即使不能直接请求这样一个查询动作，也必须取消该搜索进程。设备搜寻只寻找已经被连接的远程设备。许多蓝牙设备默认不会被搜寻到，并且需要进入到一个特殊的模式当中。

如果蓝牙状态不是 STATE_ON，这个 API 将返回 false。蓝牙打开后，等待 ACTION_STATE_CHANGED 更新成 STATE_ON，需要 BLUETOOTH_ADMIN 权限。

☑ 返回值：成功返回 true，错误返回 false。

1.3.4　BluetoothClass.Service 类

BluetoothClass.Service 类的格式如下所示。

```
public static final class BluetoothClass.Service extends Object
```

BluetoothClass.Service 类的结构如下所示。

```
java.lang.Object
android.bluetooth.BluetoothClass.Service
```

BluetoothClass.Service 类用于定义所有的服务类常量，任意 BluetoothClass 由 0 或多个服务类编码组成。BluetoothClass.Service 类中包含如下常量。

☑ int AUDIO
☑ int CAPTURE
☑ int INFORMATION
☑ int LIMITED_DISCOVERABILITY
☑ int NETWORKING
☑ int OBJECT_TRANSFER
☑ int POSITIONING
☑ int RENDER
☑ int TELEPHONY

1.3.5　BluetoothClass.Device 类

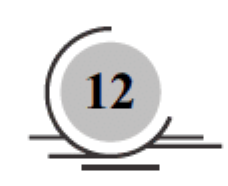
BluetoothClass.Device 类的格式如下所示。

```
public final class BluetoothClass.Device extends Object
```

BluetoothClass.Device 类的结构如下所示。

```
java.lang.Object
android.bluetooth.BluetoothClass.Device
```

BluetoothClass.Device 类用于定义所有的设备类的常量，每个 BluetoothClass 对一个带有主要和较小部分的设备类进行编码。其中的常量代表主要和较小的设备类部分（完整的设备类）的组合。



Bluetooth Class.Device.Major 的常量只能代表主要设备类，各个常量如下所示。

BluetoothClass.Device 有一个内部类，此内部类定义了所有的主要设备类常量。内部类的定义格式如下所示。

```
class BluetoothClass.Device.Major
```

注意： 到此为止，Android系统中的主要蓝牙类介绍完毕。在调用这些类时，首先确保API Level至少为版本5以上，还需注意添加相应的权限，例如在通信时需要在文件androidmanifest.xml中加入<uses-permission android:name="android.permission.BLUETOOTH" />权限，而在开关蓝牙时需要加入android.permission.BLUETOOTH_ADMIN权限。

1.4　开发一个 Android 蓝牙通信系统

知识点讲解：光盘:视频\视频讲解\第 1 章\开发一个 Android 蓝牙通信系统.avi

本实例的功能是，在 Android 手机中开发一个蓝牙通信系统，通过此系统可以实现客户端和服务器端的通信功能，下面将详细讲解本实例项目的具体实现流程。

1.4.1　主界面布局

编写布局文件，首先编写系统主界面的布局文件 main.xml，主要实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="\r\n 蓝牙测试程序\r\n"
    ></TextView>
    <Button
        android:id="@+id/startServerBtn"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="打开服务器"></Button>
    <Button
        android:id="@+id/startClientBtn"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="打开客户端"></Button>
</LinearLayout>
```

1.4.2　客户端界面布局

编写客户端的界面布局文件 client.xml，在此界面中设置了目标蓝牙设备，并提供了信息输入框，主要实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <Button
        android:id="@+id/startSearchBtn"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="开始搜索"
    ></Button>
    <TextView
        android:id="@+id/clientServersText"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dip"
        android:layout_marginBottom="10dip"
    ></TextView>
    <Button
        android:id="@+id/selectDeviceBtn"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="选择第一个设备"
    ></Button>
    <EditText
        android:id="@+id/clientChatEditText"
        android:layout_width="fill_parent"
        android:layout_height="200dip"
        android:singleLine="false"
        android:gravity="top"
        android:editable="false"
        android:cursorVisible="false"
        android:background="#aaa"
        android:layout_margin="5dip"
    ></EditText>

    <EditText
        android:id="@+id/clientSendEditText"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    ></EditText>
    <Button
        android:id="@+id/clientSendMsgBtn"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="发送"
    ></Button>
</LinearLayout>
```

服务器端的界面布局文件 server.xml 的实现过程与文件 client.xml 类似，在此不再进行详细讲解。



1.4.3　实现控制服务类和线程实现类

编写控制服务类和线程实现类，各个类的对应关系和具体结构如图 1-3 所示。

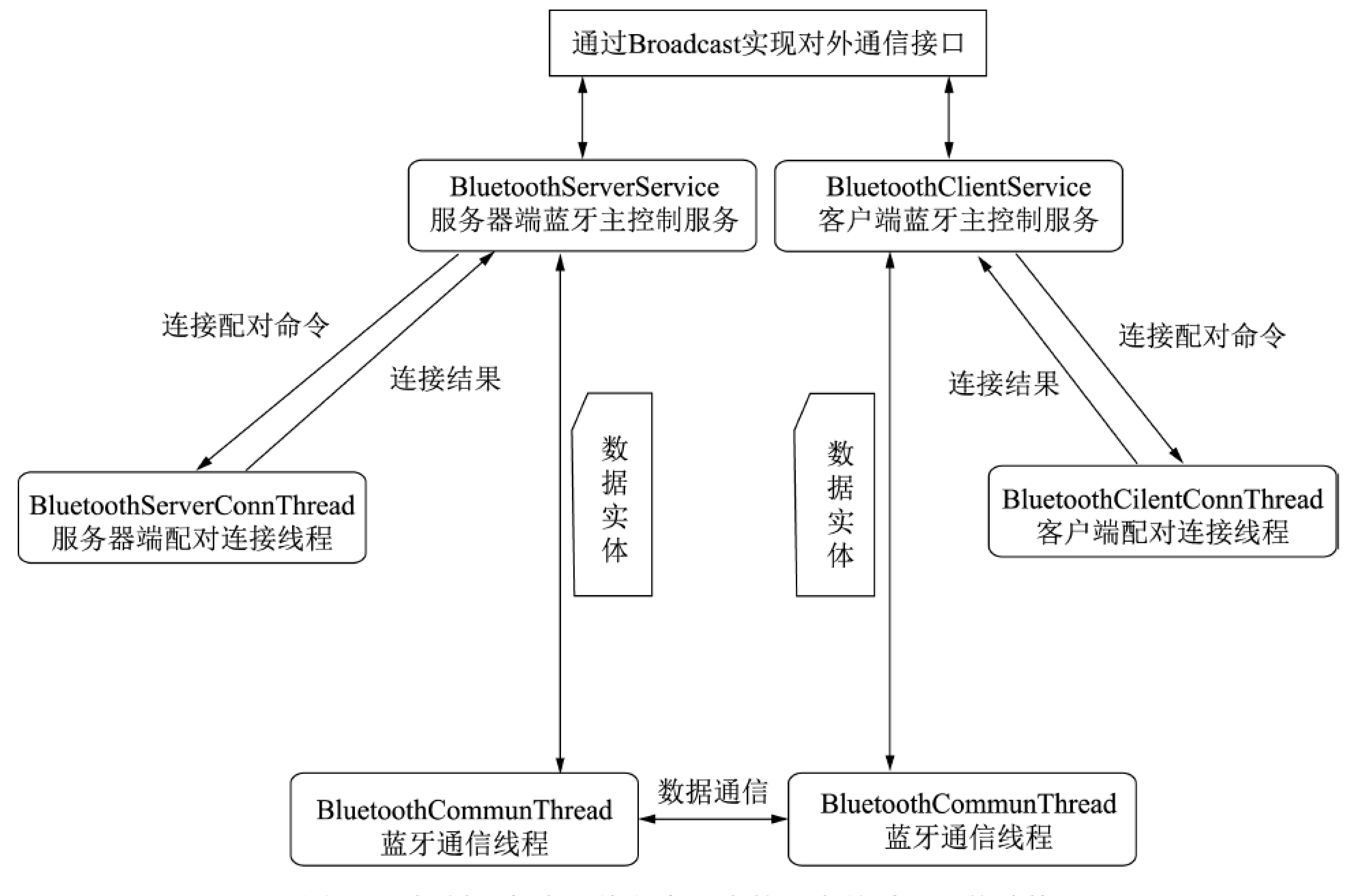


图 1-3　控制服务类和线程实现类的对应关系和具体结构

其中，BluetoothServerService 类在文件 BluetoothServerService.java 中定义，用于实现服务器端蓝牙主控制服务，此文件的主要实现代码如下所示。

```
/**
 * 蓝牙模块服务器端主控制 Service
 */
public class BluetoothServerService extends Service {
	//蓝牙适配器
	private final BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
	//蓝牙通信线程
	private BluetoothCommunThread communThread;

	//控制信息广播接收器
	private BroadcastReceiver controlReceiver = new BroadcastReceiver() {
		@Override
		public void onReceive(Context context, Intent intent) {
			String action = intent.getAction();

			if (BluetoothTools.ACTION_STOP_SERVICE.equals(action)) {
				//停止后台服务
				if (communThread != null) {
					communThread.isRun = false;
				}
				stopSelf();
```

```
                } else if (BluetoothTools.ACTION_DATA_TO_SERVICE.equals(action)) {
                    //发送数据
                    Object data = intent.getSerializableExtra(BluetoothTools.DATA);
                    if (communThread != null) {
                        communThread.writeObject(data);
                    }

                }
            }
        };
        //接收其他线程消息的 Handler
        private Handler serviceHandler = new Handler() {
            @Override
            public void handleMessage(Message msg) {

                switch (msg.what) {
                case BluetoothTools.MESSAGE_CONNECT_SUCCESS:
                    //连接成功
                    //开启通信线程
                    communThread = new BluetoothCommunThread(serviceHandler, (BluetoothSocket)msg.obj);
                    communThread.start();

                    //发送连接成功消息
                    Intent connSuccIntent = new Intent(BluetoothTools.ACTION_CONNECT_SUCCESS);
                    sendBroadcast(connSuccIntent);
                    break;

                case BluetoothTools.MESSAGE_CONNECT_ERROR:
                    //连接错误
                    //发送连接错误广播
                    Intent errorIntent = new Intent(BluetoothTools.ACTION_CONNECT_ERROR);
                    sendBroadcast(errorIntent);
                    break;
                case BluetoothTools.MESSAGE_READ_OBJECT:
                    //读取到数据
                    //发送数据广播（包含数据对象）
                    Intent dataIntent = new Intent(BluetoothTools.ACTION_DATA_TO_GAME);
                    dataIntent.putExtra(BluetoothTools.DATA, (Serializable)msg.obj);
                    sendBroadcast(dataIntent);

                    break;
                }
                super.handleMessage(msg);
            }
        };
        /**
         * 获取通信线程
         * @return
         */
        public BluetoothCommunThread getBluetoothCommunThread() {
```

```
            return communThread;
        }

        @Override
        public void onCreate() {
            //ControlReceiver 的 IntentFilter
            IntentFilter controlFilter = new IntentFilter();
            controlFilter.addAction(BluetoothTools.ACTION_START_SERVER);
            controlFilter.addAction(BluetoothTools.ACTION_STOP_SERVICE);
            controlFilter.addAction(BluetoothTools.ACTION_DATA_TO_SERVICE);

            //注册 BroadcastReceiver
            registerReceiver(controlReceiver, controlFilter);

            //开启服务器
            bluetoothAdapter.enable();        //打开蓝牙
            //开启蓝牙发现功能（300 秒）
            Intent discoveryIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
            discoveryIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, 300);
            discoveryIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(discoveryIntent);
            //开启后台连接线程
            new BluetoothServerConnThread(serviceHandler).start();
            super.onCreate();
        }
```

☑ 类 BluetoothClientService 是客户端蓝牙的主控制服务，在文件 BluetoothClientService.java 中定义，主要实现代码如下所示。

```
/**
 * 蓝牙模块客户端主控制 Service
 */
public class BluetoothClientService extends Service {

        //搜索到的远程设备集合
        private List<BluetoothDevice> discoveredDevices = new ArrayList<BluetoothDevice>();
        //蓝牙适配器
        private final BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        //蓝牙通信线程
        private BluetoothCommunThread communThread;
        //控制信息广播的接收器
        private BroadcastReceiver controlReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                String action = intent.getAction();
                if (BluetoothTools.ACTION_START_DISCOVERY.equals(action)) {
                    //开始搜索
                    discoveredDevices.clear();            //清空存放设备的集合
                    bluetoothAdapter.enable();            //打开蓝牙
                    bluetoothAdapter.startDiscovery();    //开始搜索
                } else if (BluetoothTools.ACTION_SELECTED_DEVICE.equals(action)) {
                    //选择了连接的服务器设备
```

```
                    BluetoothDevice device = (BluetoothDevice)intent.getExtras().get(BluetoothTools.DEVICE);
                    //开启设备连接线程
                    new BluetoothClientConnThread(handler, device).start();
                } else if (BluetoothTools.ACTION_STOP_SERVICE.equals(action)) {
                    //停止后台服务
                    if (communThread != null) {
                        communThread.isRun = false;
                    }
                    stopSelf();
                } else if (BluetoothTools.ACTION_DATA_TO_SERVICE.equals(action)) {
                    //获取数据
                    Object data = intent.getSerializableExtra(BluetoothTools.DATA);
                    if (communThread != null) {
                        communThread.writeObject(data);
                    }

                }
            }
        };
        //蓝牙搜索广播的接收器
        private BroadcastReceiver discoveryReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                //获取广播的 Action
                String action = intent.getAction();
                if (BluetoothAdapter.ACTION_DISCOVERY_STARTED.equals(action)) {
                    //开始搜索
                } else if (BluetoothDevice.ACTION_FOUND.equals(action)) {
                    //发现远程蓝牙设备
                    //获取设备
                    BluetoothDevice bluetoothDevice = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                    discoveredDevices.add(bluetoothDevice);
                    //发送发现设备广播
                    Intent deviceListIntent = new Intent(BluetoothTools.ACTION_FOUND_DEVICE);
                    deviceListIntent.putExtra(BluetoothTools.DEVICE, bluetoothDevice);
                    sendBroadcast(deviceListIntent);
                } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
                    //搜索结束
                    if (discoveredDevices.isEmpty()) {
                        //若未找到设备，则发动未发现设备广播
                        Intent foundIntent = new Intent(BluetoothTools.ACTION_NOT_FOUND_SERVER);
                        sendBroadcast(foundIntent);
                    }
                }
            }
        };
        //接收其他线程消息的 Handler
        Handler handler = new Handler() {
            @Override
            public void handleMessage(Message msg) {
                //处理消息
```

```
            switch (msg.what) {
            case BluetoothTools.MESSAGE_CONNECT_ERROR:
                //连接错误
                //发送连接错误广播
                Intent errorIntent = new Intent(BluetoothTools.ACTION_CONNECT_ERROR);
                sendBroadcast(errorIntent);
                break;
            case BluetoothTools.MESSAGE_CONNECT_SUCCESS:
                //连接成功

                //开启通信线程
                communThread = new BluetoothCommunThread(handler, (BluetoothSocket)msg.obj);
                communThread.start();

                //发送连接成功广播
                Intent succIntent = new Intent(BluetoothTools.ACTION_CONNECT_SUCCESS);
                sendBroadcast(succIntent);
                break;
            case BluetoothTools.MESSAGE_READ_OBJECT:
                //读取到对象
                //发送数据广播（包含数据对象）
                Intent dataIntent = new Intent(BluetoothTools.ACTION_DATA_TO_GAME);
                dataIntent.putExtra(BluetoothTools.DATA, (Serializable)msg.obj);
                sendBroadcast(dataIntent);
                break;
            }
            super.handleMessage(msg);
        }
    };
    /**
     *  获取通信线程
     * @return
     */
    public BluetoothCommunThread getBluetoothCommunThread() {
        return communThread;
    }
    @Override
    public void onStart(Intent intent, int startId) {

        super.onStart(intent, startId);
    }

    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }
    /**
     * Service 创建时的回调函数
     */
    @Override
    public void onCreate() {
```

```
        //discoveryReceiver 的 IntentFilter
        IntentFilter discoveryFilter = new IntentFilter();
        discoveryFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_STARTED);
        discoveryFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
        discoveryFilter.addAction(BluetoothDevice.ACTION_FOUND);

        //controlReceiver 的 IntentFilter
        IntentFilter controlFilter = new IntentFilter();
        controlFilter.addAction(BluetoothTools.ACTION_START_DISCOVERY);
        controlFilter.addAction(BluetoothTools.ACTION_SELECTED_DEVICE);
        controlFilter.addAction(BluetoothTools.ACTION_STOP_SERVICE);
        controlFilter.addAction(BluetoothTools.ACTION_DATA_TO_SERVICE);

        //注册 BroadcastReceiver
        registerReceiver(discoveryReceiver, discoveryFilter);
        registerReceiver(controlReceiver, controlFilter);
        super.onCreate();
    }
    /**
     * Service 销毁时的回调函数
     */
    @Override
    public void onDestroy() {
        if (communThread != null) {
            communThread.isRun = false;
        }
        //解除绑定
        unregisterReceiver(discoveryReceiver);
        super.onDestroy();
    }
}
```

☑ 类 BluetoothServerConnThread 是服务器端配对的连接线程，在文件 BluetoothServerConnThread.java 中定义，主要实现代码如下所示。

```
/**
 * 服务器连接线程
 */
public class BluetoothServerConnThread extends Thread {
    private Handler serviceHandler;          //用于同 Service 通信的 Handler
    private BluetoothAdapter adapter;
    private BluetoothSocket socket;          //用于通信的 Socket
    private BluetoothServerSocket serverSocket;

    /**
     * 构造函数
     * @param handler
     */
    public BluetoothServerConnThread(Handler handler) {
        this.serviceHandler = handler;
        adapter = BluetoothAdapter.getDefaultAdapter();
    }
```

```
	@Override
	public void run() {
		try {
			serverSocket = adapter.listenUsingRfcommWithServiceRecord("Server", BluetoothTools.PRIVATE_UUID);
			socket = serverSocket.accept();
		} catch (Exception e) {
			//发送连接失败消息
	serviceHandler.obtainMessage(BluetoothTools.MESSAGE_CONNECT_ERROR).sendToTarget();
			e.printStackTrace();
			return;
		} finally {
			try {
				serverSocket.close();
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
		if (socket != null) {
			//发送连接成功消息，消息的 obj 字段为连接的 socket
			Message msg = serviceHandler.obtainMessage();
			msg.what = BluetoothTools.MESSAGE_CONNECT_SUCCESS;
			msg.obj = socket;
			msg.sendToTarget();
		} else {
			//发送连接失败消息
	serviceHandler.obtainMessage(BluetoothTools.MESSAGE_CONNECT_ERROR).sendToTarget();
			return;
		}
	}
}
```

☑ 类 BluetoothClientConnThread 是客户端配对的连接线程，在文件 BluetoothClientConnThread.java 中定义，此文件的主要实现代码如下所示。

```
/**
 * 蓝牙客户端连接线程
 */
public class BluetoothClientConnThread extends Thread{
	private Handler serviceHandler;		//用于向客户端 Service 回传消息的 handler
	private BluetoothDevice serverDevice;	//服务器设备
	private BluetoothSocket socket;		//通信 Socket
	/**
	 * 构造函数
	 * @param handler
	 * @param serverDevice
	 */
	public BluetoothClientConnThread(Handler handler, BluetoothDevice serverDevice) {
		this.serviceHandler = handler;
		this.serverDevice = serverDevice;
	}
	@Override
	public void run() {
```

```
            BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
            try {
                socket = serverDevice.createRfcommSocketToServiceRecord(BluetoothTools.PRIVATE_UUID);
                BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
                socket.connect();

            } catch (Exception ex) {
                try {
                    socket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                //发送连接失败消息
        serviceHandler.obtainMessage(BluetoothTools.MESSAGE_CONNECT_ERROR).sendToTarget();
                return;
            }
            //发送连接成功消息，消息的 obj 参数为连接的 socket
            Message msg = serviceHandler.obtainMessage();
            msg.what = BluetoothTools.MESSAGE_CONNECT_SUCCESS;
            msg.obj = socket;
            msg.sendToTarget();
        }
}
```

☑ 类 BluetoothCommunThread 是蓝牙通信线程类，在文件 BluetoothCommunThread.java 中定义，此文件的主要实现代码如下所示。

```
/**
 * 蓝牙通信线程
 */
public class BluetoothCommunThread extends Thread {

    private Handler serviceHandler;                 //与 Service 通信的 Handler
    private BluetoothSocket socket;
    private ObjectInputStream inStream;             //对象输入流
    private ObjectOutputStream outStream;           //对象输出流
    public volatile boolean isRun = true;           //运行标志位

    /**
     * 构造函数
     * @param handler  用于接收消息
     * @param socket
     */
    public BluetoothCommunThread(Handler handler, BluetoothSocket socket) {
        this.serviceHandler = handler;
        this.socket = socket;
        try {
            this.outStream = new ObjectOutputStream(socket.getOutputStream());
            this.inStream = new ObjectInputStream(new BufferedInputStream(socket.getInputStream()));
        } catch (Exception e) {
            try {
                socket.close();
```

```
				} catch (IOException e1) {
					e1.printStackTrace();
				}
				//发送连接失败消息

serviceHandler.obtainMessage(BluetoothTools.MESSAGE_CONNECT_ERROR).sendToTarget();
				e.printStackTrace();
			}
		}

	@Override
	public void run() {
		while (true) {
			if (!isRun) {
				break;
			}
			try {
				Object obj = inStream.readObject();
				//发送成功读取到对象的消息，消息的 obj 参数为读取到的对象
				Message msg = serviceHandler.obtainMessage();
				msg.what = BluetoothTools.MESSAGE_READ_OBJECT;
				msg.obj = obj;
				msg.sendToTarget();
			} catch (Exception ex) {
				//发送连接失败消息

serviceHandler.obtainMessage(BluetoothTools.MESSAGE_CONNECT_ERROR).sendToTarget();
				ex.printStackTrace();
				return;
			}
		}

		//关闭流
		if (inStream != null) {
			try {
				inStream.close();
			} catch (IOException e) {
				e.printStackTrace();
			}
		}
		if (outStream != null) {
			try {
				outStream.close();
			} catch (IOException e) {
				e.printStackTrace();
			}
		}
		if (socket != null) {
			try {
				socket.close();
			} catch (IOException e) {
```

```
                        e.printStackTrace();
                    }
                }
            }

            /**
             * 写入一个可序列化的对象
             * @param obj
             */
            public void writeObject(Object obj) {
                try {
                    outStream.flush();
                    outStream.writeObject(obj);
                    outStream.flush();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

            }
        }
```

1.4.4 编写测试程序文件

文件 MainActivity.java 用于监听用户在主界面中的单击按钮事件，即监听是单击了“打开服务器”按钮还是单击了“打开客户端”按钮。文件 MainActivity.java 的主要实现代码如下所示。

```
class ButtonClickListener implements OnClickListener {
    @Override
    public void onClick(View arg0) {
        switch (arg0.getId()) {

        case R.id.startServerBtn:
            //打开服务器
            Intent serverIntent = new Intent(MainActivity.this, ServerActivity.class);
            serverIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(serverIntent);
            break;

        case R.id.startClientBtn:
            //打开客户端
            Intent clientIntent = new Intent(MainActivity.this, ClientActivity.class);
            clientIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(clientIntent);
            break;
        }
    }
}
```

文件 ServerActivity.java 实现了服务器端的数据接收和发送功能，主要实现代码如下所示。

```
    //广播接收器
    private BroadcastReceiver broadcastReceiver = new BroadcastReceiver() {
```

```
        @Override
        public void onReceive(Context context, Intent intent) {

            String action = intent.getAction();

            if (BluetoothTools.ACTION_DATA_TO_GAME.equals(action)) {
                //接收数据
                TransmitBean data = (TransmitBean)intent.getExtras().getSerializable(BluetoothTools.DATA);
                String msg = "from remote " + new Date().toLocaleString() + " :\r\n" + data.getMsg() + "\r\n";
                msgEditText.append(msg);

            } else if (BluetoothTools.ACTION_CONNECT_SUCCESS.equals(action)) {
                //连接成功
                serverStateTextView.setText("连接成功");
                sendBtn.setEnabled(true);
            }

        }
    };
    @Override
    protected void onStart() {
        //开启后台 service
        Intent startService = new Intent(ServerActivity.this, BluetoothServerService.class);
        startService(startService);
        //注册 BroadcastReceiver
        IntentFilter intentFilter = new IntentFilter();
        intentFilter.addAction(BluetoothTools.ACTION_DATA_TO_GAME);
        intentFilter.addAction(BluetoothTools.ACTION_CONNECT_SUCCESS);

        registerReceiver(broadcastReceiver, intentFilter);
        super.onStart();
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.server);
        serverStateTextView = (TextView)findViewById(R.id.serverStateText);
        serverStateTextView.setText("正在连接...");
        msgEditText = (EditText)findViewById(R.id.serverEditText);
        sendMsgEditText = (EditText)findViewById(R.id.serverSendEditText);
        sendBtn = (Button)findViewById(R.id.serverSendMsgBtn);
        sendBtn.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                if ("".equals(sendMsgEditText.getText().toString().trim())) {
                    Toast.makeText(ServerActivity.this, "输入不能为空", Toast.LENGTH_SHORT).show();
                } else {
                    //发送消息
                    TransmitBean data = new TransmitBean();
                    data.setMsg(sendMsgEditText.getText().toString());
```

```
                    Intent sendDataIntent = new Intent(BluetoothTools.ACTION_DATA_TO_SERVICE);
                    sendDataIntent.putExtra(BluetoothTools.DATA, data);
                    sendBroadcast(sendDataIntent);
                }
            }
        });
        sendBtn.setEnabled(false);
    }
    @Override
    protected void onStop() {
        //关闭后台 Service
        Intent startService = new Intent(BluetoothTools.ACTION_STOP_SERVICE);
        sendBroadcast(startService);

        unregisterReceiver(broadcastReceiver);
        super.onStop();
    }
```

文件 ClientActivity.java 实现了客户端的数据接收和发送功能，主要实现代码如下所示。

```
    //广播接收器
    private BroadcastReceiver broadcastReceiver = new BroadcastReceiver() {

        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();

            if (BluetoothTools.ACTION_NOT_FOUND_SERVER.equals(action)) {
                //未发现设备
                serversText.append("not found device\r\n");

            } else if (BluetoothTools.ACTION_FOUND_DEVICE.equals(action)) {
                //获取到设备对象
                BluetoothDevice device = (BluetoothDevice)intent.getExtras().get(BluetoothTools.DEVICE);
                deviceList.add(device);
                serversText.append(device.getName() + "\r\n");

            } else if (BluetoothTools.ACTION_CONNECT_SUCCESS.equals(action)) {
                //连接成功
                serversText.append("连接成功");
                sendBtn.setEnabled(true);

            } else if (BluetoothTools.ACTION_DATA_TO_GAME.equals(action)) {
                //接收数据
                TransmitBean data = (TransmitBean)intent.getExtras().getSerializable(BluetoothTools.DATA);
                String msg = "from remote " + new Date().toLocaleString() + " :\r\n" + data.getMsg() + "\r\n";
                chatEditText.append(msg);

            }
        }
    };
```

```
@Override
protected void onStart() {
    //清空设备列表
    deviceList.clear();

    //开启后台 Service
    Intent startService = new Intent(ClientActivity.this, BluetoothClientService.class);
    startService(startService);

    //注册 BroadcastReceiver
    IntentFilter intentFilter = new IntentFilter();
    intentFilter.addAction(BluetoothTools.ACTION_NOT_FOUND_SERVER);
    intentFilter.addAction(BluetoothTools.ACTION_FOUND_DEVICE);
    intentFilter.addAction(BluetoothTools.ACTION_DATA_TO_GAME);
    intentFilter.addAction(BluetoothTools.ACTION_CONNECT_SUCCESS);

    registerReceiver(broadcastReceiver, intentFilter);

    super.onStart();
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.client);

    serversText = (TextView)findViewById(R.id.clientServersText);
    chatEditText = (EditText)findViewById(R.id.clientChatEditText);
    sendEditText = (EditText)findViewById(R.id.clientSendEditText);
    sendBtn = (Button)findViewById(R.id.clientSendMsgBtn);
    startSearchBtn = (Button)findViewById(R.id.startSearchBtn);
    selectDeviceBtn = (Button)findViewById(R.id.selectDeviceBtn);

    sendBtn.setOnClickListener(new OnClickListener() {

        @Override
        public void onClick(View v) {
            //发送消息
            if ("".equals(sendEditText.getText().toString().trim())) {
                Toast.makeText(ClientActivity.this, "输入不能为空", Toast.LENGTH_SHORT).show();
            } else {
                //发送消息
                TransmitBean data = new TransmitBean();
                data.setMsg(sendEditText.getText().toString());
                Intent sendDataIntent = new Intent(BluetoothTools.ACTION_DATA_TO_SERVICE);
                sendDataIntent.putExtra(BluetoothTools.DATA, data);
                sendBroadcast(sendDataIntent);
            }
        }
    });
```

```
        startSearchBtn.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                //开始搜索
                Intent startSearchIntent = new Intent(BluetoothTools.ACTION_START_DISCOVERY);
                sendBroadcast(startSearchIntent);
            }
        });

        selectDeviceBtn.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                //选择第一个设备
                Intent selectDeviceIntent = new Intent(BluetoothTools.ACTION_SELECTED_DEVICE);
                selectDeviceIntent.putExtra(BluetoothTools.DEVICE, deviceList.get(0));
                sendBroadcast(selectDeviceIntent);
            }
        });
    }

    @Override
    protected void onStop() {
        //关闭后台 Service
        Intent startService = new Intent(BluetoothTools.ACTION_STOP_SERVICE);
        sendBroadcast(startService);

        unregisterReceiver(broadcastReceiver);
        super.onStop();
    }
```

在本部分的实现代码中，涉及蓝牙系统中的各种 Action。在 Android 系统的蓝牙通信模块中，用户通过 Broadcast（广播）与后台通信模块 Service 进行通信（控制 Service 或者接收反馈信息）。在下面列出了对应的广播 Action 的类型。

- ☑ 服务器与客户端公用 Action 列表。
  - ➢ ACTION_STOP_SERVICE：关闭后台服务。当程序退出或需要停止蓝牙服务时发送此广播。
  - ➢ ACTION_DATA_TO_SERVICE：数据传送至后台 Service。包含一个 key 为 DATA 的参数，该参数类型为实现了 Serializable 接口的类（该类为用户自己编写的数据实体类）。
  - ➢ ACTION_CONNECT_SUCCESS：连接成功。从后台 Service 发送出连接成功建立的广播。
  - ➢ ACTION_CONNECT_ERROR：连接错误。从后台 Serivce 发送出连接发生错误的广播。
  - ➢ ACTION_DATA_TO_GAME：从后台 Service 传送出数据。包含一个 key 为 DATA 的参数，该参数类型为实现了 Serializable 接口的类（该类为用户自己编写的数据实体类）。
- ☑ 客户端 Action 列表。
  - ➢ ACTION_START_DISCOVERY：开启蓝牙搜索。命令后台 Service 开始蓝牙搜索。
  - ➢ ACTION_SELECTED_DEVICE：选中的蓝牙设备。包含一个 key 为 DEVICE 的参数，该参数类型为 BluetoothDevice（蓝牙设备类）。用户需要从搜索到的蓝牙设备中选择服务器设备，选择设备后发送 Broadcast，告知后台 Service 选择的蓝牙设备。

- ACTION_FOUND_DEVICE：发现设备。后台 Service 搜索蓝牙设备过程中，每发现一个设备便会发送该 Broadcast。
- ACTION_NOT_FOUND_SERVER：未发现服务器设备。后台 Service 通过搜索并未发现可连接的蓝牙设备，并发送此 Broadcast。

到此为止，整个实例介绍完毕，执行后的效果如图 1-4 所示。



图 1-4　执行效果

客户端界面效果如图 1-5 所示，在此界面中可以实现和服务器端的数据传递功能。



图 1-5　客户端的界面效果

第 2 章　移动微信系统

微信是腾讯公司于 2011 年 1 月 21 日推出的一款通过网络快速发送语音短信、视频、图片和文字，支持多人群聊的手机聊天软件。用户可以通过微信与好友进行形式上更加丰富的类似于短信、彩信等方式的联系。微信软件本身完全免费，使用任何功能都不会收取费用，使用微信时产生的上网流量费由网络运营商收取。2012 年 9 月 17 日，微信注册用户超过 2 亿。本章将详细讲解在 Android 平台中开发一个微信系统的基本思路和具体实现流程。

2.1　微信系统基础

知识点讲解：光盘:视频\视频讲解\第 2 章\微信系统基础.avi

微信系统与平常用的 QQ 聊天软件类似，也是一款交流通信工具，能够实现在线实时交流。在本节的内容中，将简要介绍微信系统的基本知识。

2.1.1　微信的特点

微信是一种更快速的即时通信工具，具有零资费、跨平台沟通、显示实时输入状态等功能，与传统的短信沟通方式相比，更灵活、智能，且节省资费。截至 2012 年 9 月 17 日用户量已经达到了 2 亿。微信的具体特点如下：

☑ 支持发送语音短信、视频、图片（包括表情）和文字。
☑ 支持多人群聊，最高 20 人。100 人、200 人群聊功能正在内测。
☑ 支持查看所在位置附近使用微信的人（LBS 功能）。
☑ 支持腾讯微博、QQ 邮箱、漂流瓶、语音记事本、QQ 同步助手等插件功能。
☑ 支持视频聊天。
☑ 微行情：支持及时查询股票行情。
☑ 多平台，支持 iPhone、Android、Windows Phone、塞班、Blackberry 平台的手机之间相互收发消息。
☑ 省流量。

2.1.2　微信和 Q 信、腾讯的关系

Q 信是另一款腾讯手机软件 QQ 通讯录中的一个功能，与微信功能极其相似，但其实是两个不同的软件，区别如下。

（1）微信上也集成很多插件，如 QQ 邮箱助手、QQ 离线助手、通讯录安全助手等，Q 信则只是 QQ 通讯录上的一个功能，没有下层插件。

（2）微信的好友是基于手机上有电话号码的联系人和 QQ 上的好友，而 Q 信只基于手机上电话号码的联系人。

（3）Q 信能够显示好友手机号码，微信只能显示称呼或者备注名字，不能显示手机号码。

Q 信和微信虽然基本功能完全一样，可以通过网络快速发送免费（需消耗少量网络流量）语音短信、视频、图片和文字，支持多人群聊，但是是独立的两个软件，腾讯公司旗下的不同产品。开通了其中一个，另外一个还是要单独开通的。

2.2　开发一个微信系统

知识点讲解：光盘:视频\视频讲解\第 2 章\开发一个微信系统.avi

在本节的内容中，将通过一个具体实例的实现过程，详细讲解在 Android 平台中开发一个仿微信系统的基本流程。

2.2.1　启动界面

使用过微信的用户都知道，每次启动程序都会有一个启动画面，如果是第一次使用当然还会出现后面的导航界面。当本实例启动后会进入第一个 Activity，此 Activity 就是一个启动画面，之后会在这个 Activity 中设置一个 Handler 去延迟（1 秒，数值可以自己设定）执行启动导航界面的 Activity。

启动界面的 UI 文件是 appstart.xml，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="@drawable/welcome" >
</LinearLayout>
```

启动界面的实现文件是 Appstart.java，具体代码如下所示。

```
package cn.buaa.myweixin;
import android.os.Bundle;
import android.os.Handler;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.WindowManager;

public class Appstart extends Activity{

    @Override
    public void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.appstart);
        //requestWindowFeature(Window.FEATURE_NO_TITLE);     //去掉标题栏
        //getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
        //WindowManager.LayoutParams.FLAG_FULLSCREEN);      //全屏显示
        //Toast.makeText(getApplicationContext(), "孩子！好好背诵！", Toast.LENGTH_LONG).show();
        //overridePendingTransition(R.anim.hyperspace_in, R.anim.hyperspace_out);
```

```
        new Handler().postDelayed(new Runnable(){
            public void run(){
                Intent intent = new Intent (Appstart.this,Welcome.class);
                startActivity(intent);
                Appstart.this.finish();
            }
        }, 1000);
    }
}
```

执行后的效果如图 2-1 所示。



图 2-1　执行效果

2.2.2　系统导航界面

进入系统后，会在界面下方显示导航选项卡，分别显示为“微信”、“通讯录”、“朋友们”和“设置”，如图 2-2 所示。

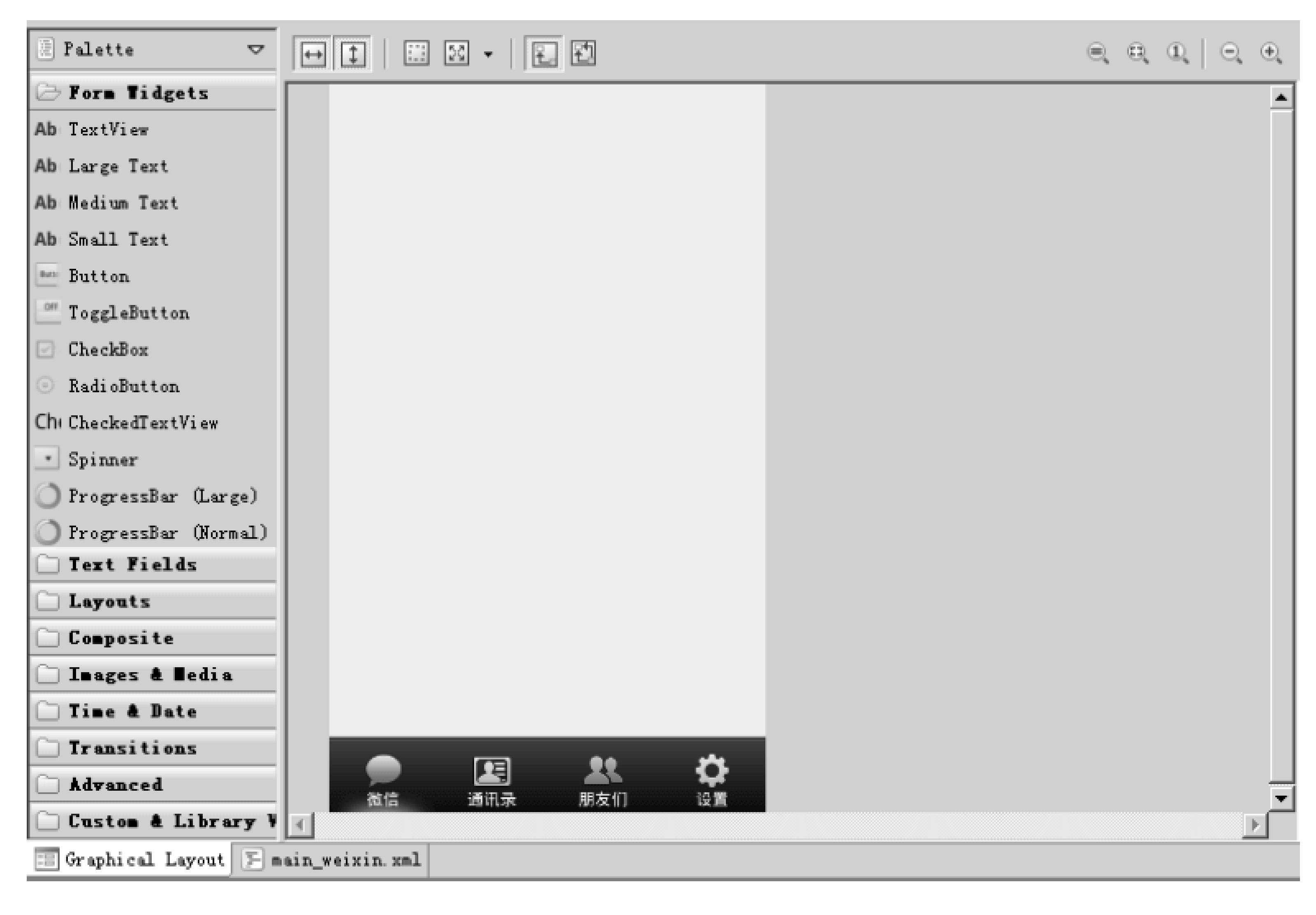


图 2-2　导航选项卡的设计界面

导航界面的布局文件是 main_weixin.xml，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mainweixin"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="#eee" >

    <RelativeLayout
        android:id="@+id/main_bottom"
        android:layout_width="match_parent"
        android:layout_height="55dp"
        android:layout_alignParentBottom="true"
        android:orientation="vertical"
        android:background="@drawable/bottom_bar"
        >

        <ImageView
            android:id="@+id/img_tab_now"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:scaleType="matrix"
            android:layout_gravity="bottom"
            android:layout_alignParentBottom="true"
            android:src="@drawable/tab_bg" />


    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:paddingBottom="2dp"
        >

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center_horizontal"
            android:orientation="vertical"
            android:layout_weight="1">
       <ImageView
              android:id="@+id/img_weixin"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:scaleType="matrix"
              android:clickable="true"
              android:src="@drawable/tab_weixin_pressed" />
              <TextView
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
```

```
                    android:text="微信"
                    android:textColor="#fff"
                    android:textSize="12sp" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:orientation="vertical"
        android:layout_weight="1">
        <ImageView
            android:id="@+id/img_address"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:scaleType="matrix"
            android:clickable="true"
            android:src="@drawable/tab_address_normal" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="通讯录"
            android:textColor="#fff"
            android:textSize="12sp" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:orientation="vertical"
        android:layout_weight="1">
    <ImageView
        android:id="@+id/img_friends"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="matrix"
        android:clickable="true"
        android:src="@drawable/tab_find_frd_normal" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="朋友们"
        android:textColor="#fff"
        android:textSize="12sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:orientation="vertical"
        android:layout_weight="1">
```

```
            <ImageView
                android:id="@+id/img_settings"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:scaleType="matrix"
                android:clickable="true"
                android:src="@drawable/tab_settings_normal" />
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="设置"
                android:textColor="#fff"
                android:textSize="12sp" />
            </LinearLayout>

        </LinearLayout>

</RelativeLayout>
<LinearLayout
android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_above="@id/main_bottom"
        android:orientation="vertical" >

<android.support.v4.view.ViewPager
        android:id="@+id/tabpager"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center" >
</android.support.v4.view.ViewPager>
</LinearLayout>
</RelativeLayout>
```

对应的实现文件是 MainWeixin.java，具体代码如下所示。

```
package cn.buaa.myweixin;
import java.util.ArrayList;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.support.v4.view.PagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v4.view.ViewPager.OnPageChangeListener;
import android.view.Display;
import android.view.Gravity;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.WindowManager;
import android.view.WindowManager.LayoutParams;
import android.view.animation.Animation;
import android.view.animation.TranslateAnimation;
```

```
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.PopupWindow;
public class MainWeixin extends Activity {
        public static MainWeixin instance = null;
        private ViewPager mTabPager;
        private ImageView mTabImg;          //动画图片
        private ImageView mTab1, mTab2, mTab3, mTab4;
        private int zero = 0;               //动画图片偏移量
        private int currIndex = 0;          //当前页卡编号
        private int one;                    //单个水平动画位移
        private int two;
        private int three;
        private LinearLayout mClose;
        private LinearLayout mCloseBtn;
        private View layout;
        private boolean menu_display = false;
        private PopupWindow menuWindow;
        private LayoutInflater inflater;

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main_weixin);
                //启动 Activity 时不自动弹出软键盘
                getWindow().setSoftInputMode(
                                WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);
                instance = this;
                mTabPager = (ViewPager) findViewById(R.id.tabpager);
                mTabPager.setOnPageChangeListener(new MyOnPageChangeListener());

                mTab1 = (ImageView) findViewById(R.id.img_weixin);
                mTab2 = (ImageView) findViewById(R.id.img_address);
                mTab3 = (ImageView) findViewById(R.id.img_friends);
                mTab4 = (ImageView) findViewById(R.id.img_settings);

                mTabImg = (ImageView) findViewById(R.id.img_tab_now);

                mTab1.setOnClickListener(new MyOnClickListener(0));
                mTab2.setOnClickListener(new MyOnClickListener(1));
                mTab3.setOnClickListener(new MyOnClickListener(2));
                mTab4.setOnClickListener(new MyOnClickListener(3));

                Display currDisplay = getWindowManager().getDefaultDisplay();//获取屏幕当前分辨率
                int displayWidth = currDisplay.getWidth();
                int displayHeight = currDisplay.getHeight();
                one = displayWidth / 4; //设置水平动画平移大小
                two = one * 2;
                three = one * 3;
                // Log.i("info", "获取的屏幕分辨率为" + one + two + three + "X" + displayHeight);
                //将要分页显示的 View 装入数组中
```

```
        LayoutInflater mLi = LayoutInflater.from(this);
        View view1 = mLi.inflate(R.layout.main_tab_weixin, null);
        View view2 = mLi.inflate(R.layout.main_tab_address, null);
        View view3 = mLi.inflate(R.layout.main_tab_friends, null);
        View view4 = mLi.inflate(R.layout.main_tab_settings, null);

        //每个页面的 View 数据
        final ArrayList<View> views = new ArrayList<View>();
        views.add(view1);
        views.add(view2);
        views.add(view3);
        views.add(view4);

        //填充 ViewPager 的数据适配器
        PagerAdapter mPagerAdapter = new PagerAdapter() {

            @Override
            public boolean isViewFromObject(View arg0, Object arg1) {
                return arg0 == arg1;
            }

            @Override
            public int getCount() {
                return views.size();
            }

            @Override
            public void destroyItem(View container, int position, Object object) {
                ((ViewPager) container).removeView(views.get(position));
            }

            // @Override
            // public CharSequence getPageTitle(int position) {
            // return titles.get(position);
            // }

            @Override
            public Object instantiateItem(View container, int position) {
                ((ViewPager) container).addView(views.get(position));
                return views.get(position);
            }
        };

        mTabPager.setAdapter(mPagerAdapter);
    }

    /**
     * 图标单击监听
     */
    public class MyOnClickListener implements View.OnClickListener {
        private int index = 0;
```

```
        public MyOnClickListener(int i) {
            index = i;
        }

        public void onClick(View v) {
            mTabPager.setCurrentItem(index);
        }
    };

    /*
     * 页卡切换监听(原作者:D.Winter)
     */
    public class MyOnPageChangeListener implements OnPageChangeListener {
        public void onPageSelected(int arg0) {
            Animation animation = null;
            switch (arg0) {
            case 0:
                mTab1.setImageDrawable(getResources().getDrawable(
                        R.drawable.tab_weixin_pressed));
                if (currIndex == 1) {
                    animation = new TranslateAnimation(one, 0, 0, 0);
                    mTab2.setImageDrawable(getResources().getDrawable(
                            R.drawable.tab_address_normal));
                } else if (currIndex == 2) {
                    animation = new TranslateAnimation(two, 0, 0, 0);
                    mTab3.setImageDrawable(getResources().getDrawable(
                            R.drawable.tab_find_frd_normal));
                } else if (currIndex == 3) {
                    animation = new TranslateAnimation(three, 0, 0, 0);
                    mTab4.setImageDrawable(getResources().getDrawable(
                            R.drawable.tab_settings_normal));
                }
                break;
            case 1:
                mTab2.setImageDrawable(getResources().getDrawable(
                        R.drawable.tab_address_pressed));
                if (currIndex == 0) {
                    animation = new TranslateAnimation(zero, one, 0, 0);
                    mTab1.setImageDrawable(getResources().getDrawable(
                            R.drawable.tab_weixin_normal));
                } else if (currIndex == 2) {
                    animation = new TranslateAnimation(two, one, 0, 0);
                    mTab3.setImageDrawable(getResources().getDrawable(
                            R.drawable.tab_find_frd_normal));
                } else if (currIndex == 3) {
                    animation = new TranslateAnimation(three, one, 0, 0);
                    mTab4.setImageDrawable(getResources().getDrawable(
                            R.drawable.tab_settings_normal));
                }
                break;
```

```
            case 2:
                mTab3.setImageDrawable(getResources().getDrawable(
                        R.drawable.tab_find_frd_pressed));
                if (currIndex == 0) {
                    animation = new TranslateAnimation(zero, two, 0, 0);
                    mTab1.setImageDrawable(getResources().getDrawable(
                            R.drawable.tab_weixin_normal));
                } else if (currIndex == 1) {
                    animation = new TranslateAnimation(one, two, 0, 0);
                    mTab2.setImageDrawable(getResources().getDrawable(
                            R.drawable.tab_address_normal));
                } else if (currIndex == 3) {
                    animation = new TranslateAnimation(three, two, 0, 0);
                    mTab4.setImageDrawable(getResources().getDrawable(
                            R.drawable.tab_settings_normal));
                }
                break;
            case 3:
                mTab4.setImageDrawable(getResources().getDrawable(
                        R.drawable.tab_settings_pressed));
                if (currIndex == 0) {
                    animation = new TranslateAnimation(zero, three, 0, 0);
                    mTab1.setImageDrawable(getResources().getDrawable(
                            R.drawable.tab_weixin_normal));
                } else if (currIndex == 1) {
                    animation = new TranslateAnimation(one, three, 0, 0);
                    mTab2.setImageDrawable(getResources().getDrawable(
                            R.drawable.tab_address_normal));
                } else if (currIndex == 2) {
                    animation = new TranslateAnimation(two, three, 0, 0);
                    mTab3.setImageDrawable(getResources().getDrawable(
                            R.drawable.tab_find_frd_normal));
                }
                break;
            }
            currIndex = arg0;
            animation.setFillAfter(true);//True:图片停在动画结束位置
            animation.setDuration(150);//动画持续时间
            mTabImg.startAnimation(animation);//开始动画
        }

        public void onPageScrolled(int arg0, float arg1, int arg2) {
        }

        public void onPageScrollStateChanged(int arg0) {
        }
    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_BACK && event.getRepeatCount() == 0) { //获取//Back 键
```

```
        if (menu_display) { //如果 Menu 已经打开，先关闭 Menu
            menuWindow.dismiss();
            menu_display = false;
        } else {
            Intent intent = new Intent();
            intent.setClass(MainWeixin.this, Exit.class);
            startActivity(intent);
        }
    }

    else if (keyCode == KeyEvent.KEYCODE_MENU) { //获取 Menu 键
        if (!menu_display) {
            //获取 LayoutInflater 实例
            inflater = (LayoutInflater) this
            .getSystemService(LAYOUT_INFLATER_SERVICE);
            //这里的 main 布局是在 inflate 中加入的，以前都是直接调用 this.setContentView()
            // 该方法返回的是一个 View 的对象，是布局中的根
            layout = inflater.inflate(R.layout.main_menu, null);
            menuWindow = new PopupWindow(layout, LayoutParams.FILL_PARENT,
                    LayoutParams.WRAP_CONTENT); //后两个参数是 width 和 height
            // menuWindow.showAsDropDown(layout); //设置弹出效果
            // menuWindow.showAsDropDown(null, 0, layout.getHeight());
            menuWindow.showAtLocation(this.findViewById(R.id.mainweixin),
                    Gravity.BOTTOM | Gravity.CENTER_HORIZONTAL, 0, 0); //设置 layout
                    在 PopupWindow 中显示的位置
            //获取 main 中的控件
            mClose = (LinearLayout) layout.findViewById(R.id.menu_close);
            mCloseBtn = (LinearLayout) layout
                    .findViewById(R.id.menu_close_btn);

            //下面对每一个 Layout 进行单击事件的注册
            //例如，单击某个 MenuItem 时，其背景色改变
            //事先准备好一些背景图片或者颜色
            mCloseBtn.setOnClickListener(new View.OnClickListener() {
                public void onClick(View arg0) {
                    // Toast.makeText(Main.this, "退出",
                    // Toast.LENGTH_LONG).show();
                    Intent intent = new Intent();
                    intent.setClass(MainWeixin.this, Exit.class);
                    startActivity(intent);
                    menuWindow.dismiss(); //响应单击事件之后关闭 Menu
                }
            });
            menu_display = true;
        } else {
            //如果当前已经为显示状态，则隐藏起来
            menuWindow.dismiss();
            menu_display = false;
        }
```

```
                return false;
            }
            return false;
    }

    //设置标题栏右侧按钮的作用
    public void btnmainright(View v) {
            Intent intent = new Intent(MainWeixin.this, MainTopRightDialog.class);
            startActivity(intent);
            // Toast.makeText(getApplicationContext(), "单击了功能按钮",
            // Toast.LENGTH_LONG).show();
    }
    public void startchat(View v) {//小黑对话界面
            Intent intent = new Intent(MainWeixin.this, ChatActivity.class);
            startActivity(intent);
            // Toast.makeText(getApplicationContext(), "登录成功",
            // Toast.LENGTH_LONG).show();
    }

    public void exit_settings(View v) {//退出伪对话框，其实是一个 Activity
            Intent intent = new Intent(MainWeixin.this, ExitFromSettings.class);
            startActivity(intent);
    }
    public void btn_shake(View v) {//手机摇一摇
            Intent intent = new Intent(MainWeixin.this, ShakeActivity.class);
            startActivity(intent);
    }
}
```

2.2.3　系统登录界面

为了保证系统的安全，设置只有合法用户才能登录系统，为此专门设置了一个登录表单界面。具体 UI 界面如图 2-3 所示。



图 2-3　系统登录表单的设计界面

系统登录界面的布局文件是 login.xml，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#eee"
android:orientation="vertical"
android:gravity="center_horizontal">
    <RelativeLayout
        android:id="@+id/login_top_layout"
        android:layout_width="fill_parent"
        android:layout_height="45dp"
        android:layout_alignParentTop="true"
        android:background="@drawable/title_bar">
    <Button
            android:id="@+id/login_reback_btn"
            android:layout_width="70dp"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:text="返回"
            android:textSize="26sp"
            android:textColor="#fff"
            android:onClick="login_back"
            android:background="@drawable/title_btn_back"/>
    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:textSize="20sp"
            android:textStyle="bold"
            android:textColor="#ffffff"
            android:text="登录"
           />

</RelativeLayout>
     <EditText
        android:id="@+id/login_user_edit"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/login_top_layout"
        android:textColor="#000"
        android:textSize="15sp"
        android:layout_marginTop="25dp"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:singleLine="true"
        android:background="@drawable/login_editbox"
        android:hint="QQ 号/微信号/手机号（请输入 buaa）"/>
     <EditText
        android:id="@+id/login_passwd_edit"
        android:layout_width="fill_parent"
```

```
            android:layout_height="wrap_content"
            android:layout_below="@+id/login_user_edit"
            android:textColor="#000"
            android:textSize="15sp"
            android:layout_marginTop="25dp"
            android:layout_marginLeft="20dp"
            android:layout_marginRight="20dp"
            android:background="@drawable/login_editbox"
            android:password="true"
            android:singleLine="true"
            android:hint="密码(请输入 123)"/>
        <RelativeLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:layout_below="@+id/login_passwd_edit"
        >
        <Button
            android:id="@+id/forget_passwd"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="23dp"
            android:layout_marginTop="5dp"
            android:text="忘记密码?"
            android:textSize="16sp"
            android:textColor="#00f"
            android:background="#0000"
            android:onClick="login_pw"
                />
        <Button
            android:id="@+id/login_login_btn"
            android:layout_width="90dp"
            android:layout_height="40dp"
            android:layout_marginRight="20dp"
            android:layout_alignParentRight="true"
            android:text="登录"
            android:background="@drawable/btn_style_green"
            android:textColor="#ffffff"
            android:textSize="18sp"
            android:onClick="login_mainweixin"
                />
        </RelativeLayout>
</RelativeLayout>
```

对应的实现文件是 login.java，具体代码如下所示。

```
package cn.buaa.myweixin;
import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.Intent;
import android.view.Menu;
```

```
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class Login extends Activity {
     private EditText mUser;              //账号编辑框
     private EditText mPassword;          //密码编辑框

    @Override
    public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.login);

    mUser = (EditText)findViewById(R.id.login_user_edit);
    mPassword = (EditText)findViewById(R.id.login_passwd_edit);

    }
    public void login_mainweixin(View v) {
     if("weixin".equals(mUser.getText().toString()) && "123".equals(mPassword.getText().toString()))     //判断
账号和密码
        {
             Intent intent = new Intent();
             intent.setClass(Login.this,LoadingActivity.class);
             startActivity(intent);
        }
        else if("".equals(mUser.getText().toString()) || "".equals(mPassword.getText().toString()))     //判断账号
和密码
        {
              new AlertDialog.Builder(Login.this)
              .setIcon(getResources().getDrawable(R.drawable.login_error_icon))
              .setTitle("登录错误")
              .setMessage("微信账号或者密码不能为空，\n 请输入后再登录！")
              .create().show();
         }
         else{

              new AlertDialog.Builder(Login.this)
              .setIcon(getResources().getDrawable(R.drawable.login_error_icon))
              .setTitle("登录失败")
              .setMessage("微信账号或者密码不正确，\n 请检查后重新输入！")
              .create().show();
        }

     //登录按钮
     /*
     Intent intent = new Intent();
          intent.setClass(Login.this,Whatsnew.class);
          startActivity(intent);
          Toast.makeText(getApplicationContext(), "登录成功", Toast.LENGTH_SHORT).show();
```

```
            this.finish();*/
        }
      public void login_back(View v) {        //标题栏返回按钮
       this.finish();
      }
      public void login_pw(View v) {        //忘记密码按钮
       Uri uri = Uri.parse("http://3g.qq.com");
       Intent intent = new Intent(Intent.ACTION_VIEW, uri);
       startActivity(intent);
       //Intent intent = new Intent();
       //intent.setClass(Login.this,Whatsnew.class);
           //startActivity(intent);
        }
}
```

登录成功后调用文件 LoadingActivity.java 进入系统主界面，此文件的实现代码如下所示。

```
package cn.buaa.myweixin;
import android.os.Bundle;
import android.os.Handler;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.WindowManager;
import android.widget.Toast;

public class LoadingActivity extends Activity{

    @Override
    public void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.loading);

        new Handler().postDelayed(new Runnable(){
                public void run(){
                    Intent intent = new Intent (LoadingActivity.this,Whatsnew.class);
                    startActivity(intent);
                    LoadingActivity.this.finish();
                    Toast.makeText(getApplicationContext(), "登录成功", Toast.LENGTH_SHORT).show();
        }
    }, 200);
  }
}
```

2.2.4　发送信息界面

为了达到在线交流效果，系统提供了发送信息界面，此界面和 QQ 聊天界面类似，调用输入法输入文本信息。具体 UI 界面如图 2-4 所示。

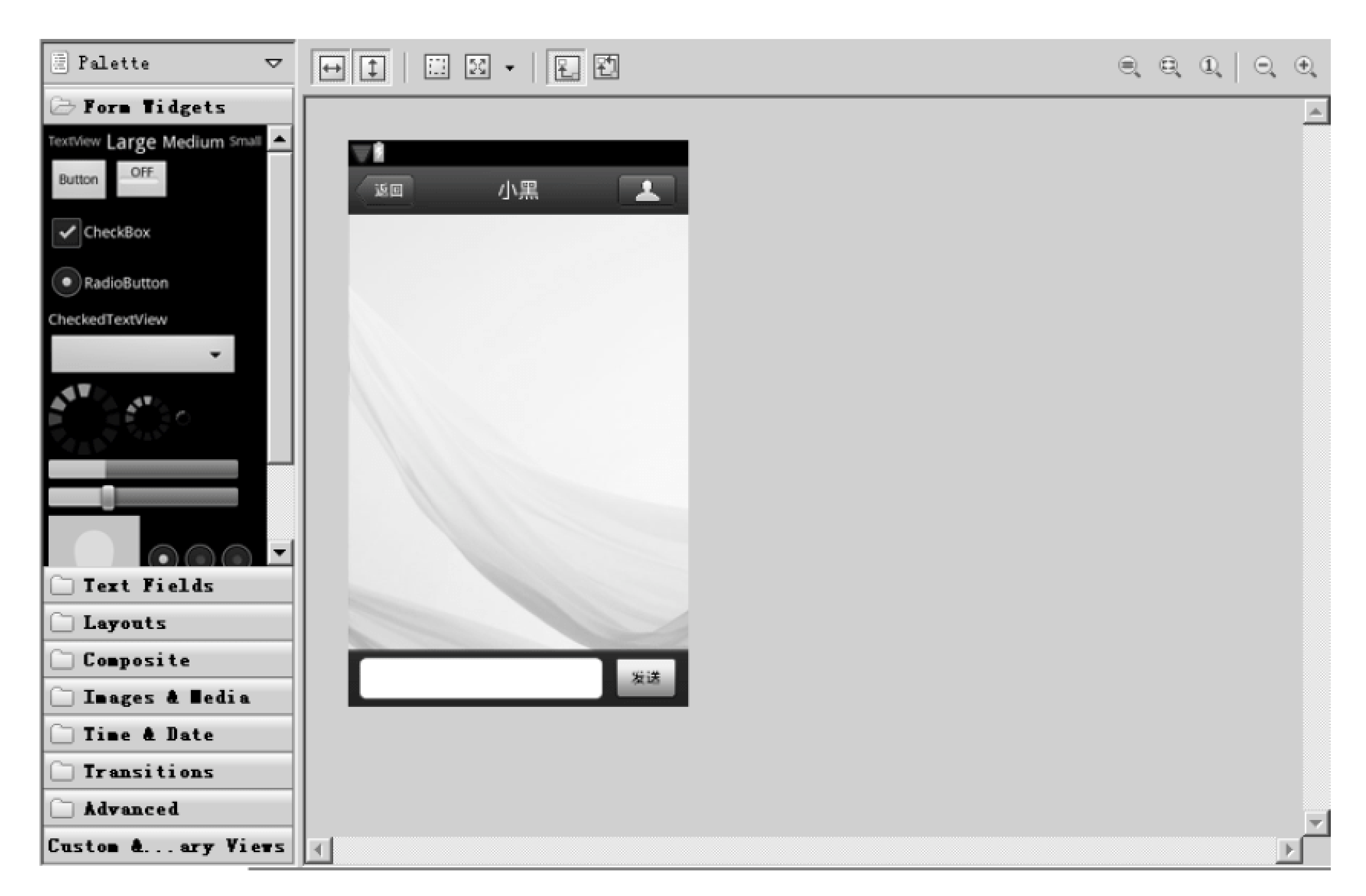


图 2-4　发送信息设计界面

系统信息发送界面的布局文件是 chat_xiaohei.xml，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
  android:background="@drawable/chat_bg_default" >

<RelativeLayout
        android:id="@+id/rl_layout"
            android:layout_width="fill_parent"
            android:layout_height="45dp"
            android:background="@drawable/title_bar"
            android:gravity="center_vertical"   >
        <Button
            android:id="@+id/btn_back"
            android:layout_width="70dp"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:text="返回"
            android:textSize="26sp"
            android:textColor="#fff"
            android:onClick="chat_back"
            android:background="@drawable/title_btn_back"
            />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="小黑"
            android:layout_centerInParent="true"
```

```
                    android:textSize="20sp"
                    android:textColor="#ffffff" />
               <ImageButton
                    android:id="@+id/right_btn"
                    android:layout_width="67dp"
                    android:layout_height="wrap_content"
                    android:layout_alignParentRight="true"
                    android:layout_centerVertical="true"
                    android:layout_marginRight="5dp"
                    android:src="@drawable/mm_title_btn_contact_normal"
                    android:background="@drawable/title_btn_right"
                    />
          </RelativeLayout>


<RelativeLayout
        android:id="@+id/rl_bottom"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:background="@drawable/chat_footer_bg" >

    <Button
        android:id="@+id/btn_send"
        android:layout_width="60dp"
        android:layout_height="40dp"
        android:layout_alignParentRight="true"
        android:layout_marginRight="10dp"
        android:layout_centerVertical="true"
        android:text="发送"
        android:background="@drawable/chat_send_btn" />


    <EditText
        android:id="@+id/et_sendmessage"
        android:layout_width="fill_parent"
        android:layout_height="40dp"
        android:layout_toLeftOf="@id/btn_send"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:background="@drawable/login_edit_normal"
        android:layout_centerVertical="true"
        android:singleLine="true"
        android:textSize="18sp"/>

</RelativeLayout>


    <ListView
        android:id="@+id/listview"
        android:layout_below="@id/rl_layout"
```

```
        android:layout_above="@id/rl_bottom"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:divider="@null"
        android:dividerHeight="5dp"
        android:stackFromBottom="true"
        android:scrollbarStyle="outsideOverlay"
        android:cacheColorHint="#0000"/>

</RelativeLayout>
```

对应的实现文件是 ChatActivity.java，具体代码如下所示。

```
package cn.buaa.myweixin;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import android.app.Activity;
import android.content.Intent;
import android.graphics.drawable.LevelListDrawable;
import android.os.Bundle;
import android.text.Editable;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;



public class ChatActivity extends Activity implements OnClickListener{
    /** Called when the activity is first created*/

     private Button mBtnSend;
     private Button mBtnBack;
     private EditText mEditTextContent;
     private ListView mListView;
     private ChatMsgViewAdapter mAdapter;
     private List<ChatMsgEntity> mDataArrays = new ArrayList<ChatMsgEntity>();


     public void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              setContentView(R.layout.chat_xiaohei);
        //启动 Activity 时不自动弹出软键盘
     getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);
     initView();

     initData();
    }
```

```
public void initView()
      {
       mListView = (ListView) findViewById(R.id.listview);
       mBtnSend = (Button) findViewById(R.id.btn_send);
       mBtnSend.setOnClickListener(this);
       mBtnBack = (Button) findViewById(R.id.btn_back);
       mBtnBack.setOnClickListener(this);

       mEditTextContent = (EditText) findViewById(R.id.et_sendmessage);
      }

      private String[ ]msgArray = new String[ ]{"有大", "有！？", "我也有", "那上吧",
                                                "打啊！你放大啊", "你不？留人头那！",
                                                "不解释", "...",};

private String[ ]dataArray = new String[ ]{"2012-09-01 18:00", "2012-09-01 18:10",
                                           "2012-09-01 18:11", "2012-09-01 18:20",
                                           "2012-09-01 18:30", "2012-09-01 18:35",
                                           "2012-09-01 18:40", "2012-09-01 18:50"};
private final static int COUNT = 8;
public void initData()
      {
       for(int i = 0; i < COUNT; i++)
       {
              ChatMsgEntity entity = new ChatMsgEntity();
              entity.setDate(dataArray[i]);
              if (i % 2 == 0)
              {
                     entity.setName("小黑");
                     entity.setMsgType(true);
              }else{
                     entity.setName("人马");
                     entity.setMsgType(false);
              }

              entity.setText(msgArray[i]);
              mDataArrays.add(entity);
       }

       mAdapter = new ChatMsgViewAdapter(this, mDataArrays);
              mListView.setAdapter(mAdapter);

      }


       public void onClick(View v) {
              // TODO Auto-generated method stub
              switch(v.getId())
              {
              case R.id.btn_send:
```

```
                send();
                break;
            case R.id.btn_back:
                finish();
                break;
            }
        }

        private void send()
        {
            String contString = mEditTextContent.getText().toString();
            if (contString.length() > 0)
            {
                ChatMsgEntity entity = new ChatMsgEntity();
                entity.setDate(getDate());
                entity.setName("人马");
                entity.setMsgType(false);
                entity.setText(contString);

                mDataArrays.add(entity);
                mAdapter.notifyDataSetChanged();

                mEditTextContent.setText("");

                mListView.setSelection(mListView.getCount() - 1);
            }
        }

private String getDate() {
            Calendar c = Calendar.getInstance();

            String year = String.valueOf(c.get(Calendar.YEAR));
            String month = String.valueOf(c.get(Calendar.MONTH));
            String day = String.valueOf(c.get(Calendar.DAY_OF_MONTH) + 1);
            String hour = String.valueOf(c.get(Calendar.HOUR_OF_DAY));
            String mins = String.valueOf(c.get(Calendar.MINUTE));


            StringBuffer sbBuffer = new StringBuffer();
            sbBuffer.append(year + "-" + month + "-" + day + " " + hour + ":" + mins);


            return sbBuffer.toString();
        }


        public void head_xiaohei(View v) {       //标题栏返回按钮
         Intent intent = new Intent (ChatActivity.this,InfoXiaohei.class);
            startActivity(intent);
          }
}
```

2.2.5　“摇一摇”界面

“摇一摇”是微信的特色功能，通过摇动手机的方式可以实现一个操作功能，例如，发送一幅图片，查找到一个好友等。具体 UI 界面如图 2-5 所示。

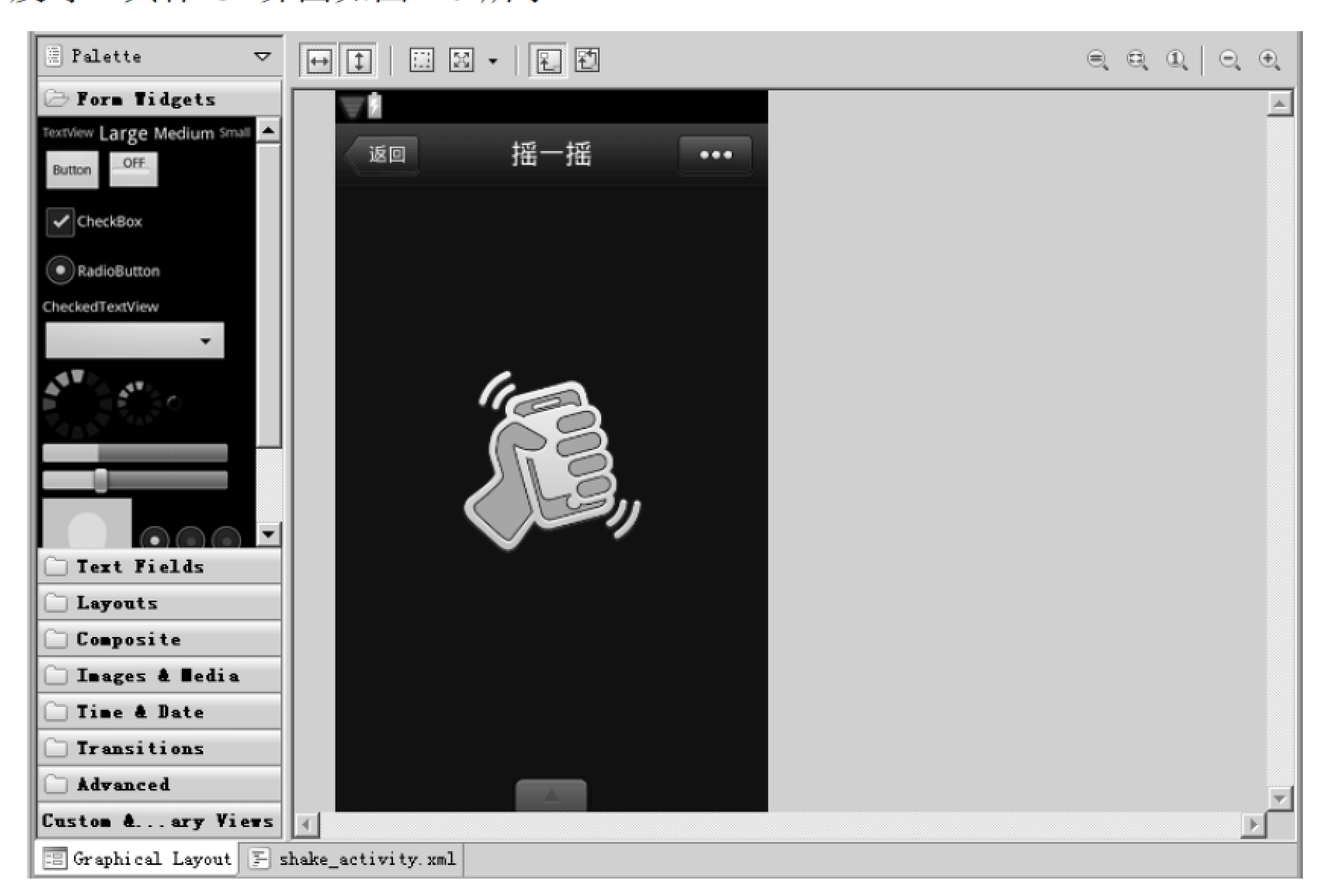


图 2-5　“摇一摇”界面

系统信息发送界面的布局文件是 shake_activity.xml，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical"
        android:background="#111"
>

<RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_centerInParent="true" >

<ImageView
        android:id="@+id/shakeBg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:src="@drawable/shakehideimg_man2" />

<LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```

```
            android:layout_centerInParent="true"
            android:orientation="vertical" >

<RelativeLayout
            android:id="@+id/shakeImgUp"
            android:layout_width="fill_parent"
            android:layout_height="190dp"
            android:background="#111">
<ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_centerHorizontal="true"
            android:src="@drawable/shake_logo_up"
             />
</RelativeLayout>
<RelativeLayout
            android:id="@+id/shakeImgDown"
            android:layout_width="fill_parent"
            android:layout_height="190dp"
            android:background="#111">
<ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:src="@drawable/shake_logo_down"
             />
</RelativeLayout>
</LinearLayout>
</RelativeLayout>

<RelativeLayout
            android:id="@+id/shake_title_bar"
            android:layout_width="fill_parent"
            android:layout_height="45dp"
            android:background="@drawable/title_bar"
            android:gravity="center_vertical"    >
                <Button
            android:layout_width="70dp"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:text="返回"
            android:textSize="26sp"
            android:textColor="#fff"
            android:onClick="shake_activity_back"
            android:background="@drawable/title_btn_back"/>
            <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="摇一摇"
            android:layout_centerInParent="true"
```

```
            android:textSize="20sp"
                android:textColor="#ffffff" />
                <ImageButton
                android:layout_width="67dp"
                android:layout_height="wrap_content"
                android:layout_alignParentRight="true"
                android:layout_centerVertical="true"
                android:layout_marginRight="5dp"
                android:src="@drawable/mm_title_btn_menu"
                android:background="@drawable/title_btn_right"
                android:onClick="linshi"
                    />
        </RelativeLayout>


<SlidingDrawer
            android:id="@+id/slidingDrawer1"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:content="@+id/content"
            android:handle="@+id/handle" >
<Button
            android:id="@+id/handle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"

            android:background="@drawable/shake_report_dragger_up" />
<LinearLayout
            android:id="@+id/content"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="#f9f9f9" >
<ImageView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:scaleType="fitXY"
            android:src="@drawable/shake_line_up" />
</LinearLayout>
</SlidingDrawer>

</RelativeLayout>
```

对应的实现文件是 ShakeActivity.java，具体代码如下所示。

```
public class ShakeActivity extends Activity{
        ShakeListener mShakeListener = null;
        Vibrator mVibrator;
        private RelativeLayout mImgUp;
        private RelativeLayout mImgDn;
        private RelativeLayout mTitle;

        private SlidingDrawer mDrawer;
        private Button mDrawerBtn;
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.shake_activity);
        //drawerSet ();//设置 drawer 监听，切换按钮的方向

        mVibrator = (Vibrator)getApplication().getSystemService(VIBRATOR_SERVICE);

        mImgUp = (RelativeLayout) findViewById(R.id.shakeImgUp);
        mImgDn = (RelativeLayout) findViewById(R.id.shakeImgDown);
        mTitle = (RelativeLayout) findViewById(R.id.shake_title_bar);

        mDrawer = (SlidingDrawer) findViewById(R.id.slidingDrawer1);
        mDrawerBtn = (Button) findViewById(R.id.handle);
        mDrawer.setOnDrawerOpenListener(new OnDrawerOpenListener()
        {    public void onDrawerOpened()
             {

    mDrawerBtn.setBackgroundDrawable(getResources().getDrawable(R.drawable.shake_report_dragger_down));
                 TranslateAnimation titleup = new TranslateAnimation(Animation.RELATIVE_TO_SELF,0f,
Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,-1.0f);
                 titleup.setDuration(200);
                 titleup.setFillAfter(true);
                 mTitle.startAnimation(titleup);
             }
        });
         /*设定 SlidingDrawer 被关闭的事件处理*/
        mDrawer.setOnDrawerCloseListener(new OnDrawerCloseListener()
        {    public void onDrawerClosed()
             {

    mDrawerBtn.setBackgroundDrawable(getResources().getDrawable(R.drawable.shake_report_dragger_up));
                 TranslateAnimation titledn = new TranslateAnimation(Animation.RELATIVE_TO_SELF,0f,
Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,-1.0f,Animation.RELATIVE_TO_SELF,0f);
                 titledn.setDuration(200);
                 titledn.setFillAfter(false);
                 mTitle.startAnimation(titledn);
             }
        });

        mShakeListener = new ShakeListener(this);
        mShakeListener.setOnShakeListener(new OnShakeListener() {
             public void onShake() {
                 //Toast.makeText(getApplicationContext(), "抱歉，暂时没有找到在同一时刻摇一摇的人。
\n 再试一次吧！", Toast.LENGTH_SHORT).show();
                 startAnim();  //开始“摇一摇”手掌动画
                 mShakeListener.stop();
                 startVibrato(); //开始震动
                 new Handler().postDelayed(new Runnable(){
                     public void run(){
```

```
                                        //Toast.makeText(getApplicationContext(), "抱歉，暂时没有找到\n 在同一时刻
                                        摇一摇的人。\n 再试一次吧！", 500).setGravity(Gravity.CENTER,0,0).show();
                                        Toast mtoast;
                                        mtoast = Toast.makeText(getApplicationContext(),
                                                  "抱歉，暂时没有找到\n 在同一时刻摇一摇的人。\n 再试一次吧！", 10);
                                             //mtoast.setGravity(Gravity.CENTER, 0, 0);
                                            mtoast.show();
                                            mVibrator.cancel();
                                            mShakeListener.start();
                                }
                           }, 2000);
                      }
                 });
    }
     public void startAnim () {     //定义摇一摇动画
          AnimationSet animup = new AnimationSet(true);
          TranslateAnimation mytranslateanimup0 = new TranslateAnimation(Animation.RELATIVE_TO_SELF,0f,
Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,-0.5f);
          mytranslateanimup0.setDuration(1000);
          TranslateAnimation mytranslateanimup1 = new TranslateAnimation(Animation.RELATIVE_TO_SELF,0f,
Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,+0.5f);
          mytranslateanimup1.setDuration(1000);
          mytranslateanimup1.setStartOffset(1000);
          animup.addAnimation(mytranslateanimup0);
          animup.addAnimation(mytranslateanimup1);
          mImgUp.startAnimation(animup);

          AnimationSet animdn = new AnimationSet(true);
          TranslateAnimation mytranslateanimdn0 = new TranslateAnimation(Animation.RELATIVE_TO_SELF,0f,
Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,+0.5f);
          mytranslateanimdn0.setDuration(1000);
          TranslateAnimation mytranslateanimdn1 = new TranslateAnimation(Animation.RELATIVE_TO_SELF,0f,
Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,0f,Animation.RELATIVE_TO_SELF,-0.5f);
          mytranslateanimdn1.setDuration(1000);
          mytranslateanimdn1.setStartOffset(1000);
          animdn.addAnimation(mytranslateanimdn0);
          animdn.addAnimation(mytranslateanimdn1);
          mImgDn.startAnimation(animdn);
     }
     public void startVibrato(){                                        //定义震动
          mVibrator.vibrate( new long[ ]{500,200,500,200}, -1); //第一个{ }里面是节奏数组，第二个参数是重复
次数，-1 为不重复，非-1 则从 pattern 的指定下标开始重复
     }

     public void shake_activity_back(View v) {                          //标题栏返回按钮
     this.finish();
     }
     public void linshi(View v) {                                       //标题栏
          startAnim();
     }
     @Override
```

```
	protected void onDestroy() {
		super.onDestroy();
		if (mShakeListener != null) {
			mShakeListener.stop();
		}
	}
}
```

文件 ShakeListener.java 的功能是通过重力感应器实现重力监听，这是实现“摇一摇”功能的基础。文件 ShakeListener.java 的具体实现代码如下所示。

```
package cn.buaa.myweixin;
import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.util.Log;

/**
 * 一个检测手机摇晃的监听器
 */
public class ShakeListener implements SensorEventListener {
	//速度阈值，当摇晃速度达到该值后产生作用
	private static final int SPEED_SHRESHOLD = 3000;
	//两次检测的时间间隔
	private static final int UPTATE_INTERVAL_TIME = 70;
	//传感器管理器
	private SensorManager sensorManager;
	//传感器
	private Sensor sensor;
	//重力感应监听器
	private OnShakeListener onShakeListener;
	//上下文
	private Context mContext;
	//手机上一个位置时重力感应坐标
	private float lastX;
	private float lastY;
	private float lastZ;
	//上次检测时间
	private long lastUpdateTime;

	//构造器
	public ShakeListener(Context c) {
		//获得监听对象
		mContext = c;
		start();
	}

	//开始
	public void start() {
		//获得传感器管理器
```

```
        sensorManager = (SensorManager) mContext
                .getSystemService(Context.SENSOR_SERVICE);
        if (sensorManager != null) {
            //获得重力传感器
            sensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        }
        //注册
        if (sensor != null) {
            sensorManager.registerListener(this, sensor,
                    SensorManager.SENSOR_DELAY_GAME);
        }

    }

    //停止检测
    public void stop() {
        sensorManager.unregisterListener(this);
    }

    //设置重力感应监听器
    public void setOnShakeListener(OnShakeListener listener) {
        onShakeListener = listener;
    }

    //重力感应器感应获得变化数据
    public void onSensorChanged(SensorEvent event) {
        //现在检测时间
        long currentUpdateTime = System.currentTimeMillis();
        //两次检测的时间间隔
        long timeInterval = currentUpdateTime - lastUpdateTime;
        //判断是否达到了检测时间间隔
        if (timeInterval < UPTATE_INTERVAL_TIME)
            return;
        //现在的时间变成 last 时间
        lastUpdateTime = currentUpdateTime;

        //获得 x，y，z 坐标
        float x = event.values[0];
        float y = event.values[1];
        float z = event.values[2];

        //获得 x，y，z 的变化值
        float deltaX = x - lastX;
        float deltaY = y - lastY;
        float deltaZ = z - lastZ;

        //将现在的坐标变成 last 坐标
        lastX = x;
        lastY = y;
        lastZ = z;
```

```
            double speed = Math.sqrt(deltaX * deltaX + deltaY * deltaY + deltaZ * deltaZ)
                        / timeInterval * 10000;
            Log.v("thelog", "===========log===================");
            //达到速度阈值，发出提示
            if (speed >= SPEED_SHRESHOLD) {
                onShakeListener.onShake();
            }
        }

        public void onAccuracyChanged(Sensor sensor, int accuracy) {

        }

        //摇晃监听接口
        public interface OnShakeListener {
            public void onShake();
        }
}
```

第 3 章　移动邮件系统

现代社会科技发展迅速，移动设备在日常生活中应用得非常广泛，主要运用于娱乐、办公、科研等多个领域。基于移动设备上的应用系统发展也相当迅速，移动设备运用领域的扩大也提高了对系统的要求。Android 作为一个开源的系统，对移动设备市场的发展提供了机遇。在本章的内容中，将详细讲解在 Android 系统中开发一个邮件系统的具体流程。

3.1　项 目 介 绍

知识点讲解：光盘:视频\视频讲解\第 3 章\项目介绍.avi

本章的邮件系统实例采用 Android 开源系统技术，利用 Java 语言和 Eclipse 开发工具对邮件系统进行开发。同时给出详细的系统设计流程、部分界面图及主要功能效果流程图。在讲解具体编码之前，先简要介绍本项目的产生背景和项目意义，为读者进行后面的系统设计及编码工作做好准备。

3.1.1　项目背景介绍

随着科学技术的发展和社会的竞争，为提高工作效率，使用邮件系统收发邮件是工作中必不可少的组成部分。使用手机或者是便捷设备在旅行、出差时处理工作事务或与朋友联系越来越流行。因此，人们的生活越来越离不开手机的陪伴。随着手机硬件和软件系统的发展，人们对移动电子设备的硬件性能和软件性能要求也越来越高。

全球使用最多的移动设备是手机，其发展十分迅速，同时手机操作系统也出现了不同种类，现在的市场上主要有 3 个手机操作系统：微软的 Windows Phone、苹果的 iOS 和谷歌的 Android 操作系统，其中只有 Android 开放源代码。全球针对 Android 平台开发的团体或个人数量庞大，因此，Android 系统得以飞速发展。既然手机如此智能，那么通过手机接收邮件可以实现吗？答案是肯定的！谷歌 Android 系统可以满足此要求。本章讲解的邮件系统实例就是基于谷歌 Android 手机平台开发的。

开发一个邮件系统，要了解邮件系统支持的通信协议及各协议之间存在什么差异，还要对开发平台有较深入的了解，分析现在流行的邮件系统的优点、缺点和用户最常用的功能。

3.1.2　项目目的

当今社会竞争激烈，工作效率非常重要，而互联网办公是其中最好的提高工作效率的方式之一，本项目的目的是开发一个在手机或者是移动设备上使用的邮件系统，该系统的主要功能是邮箱类型设定、邮件收取设置、邮件发送设置、用户检查、用户别名设置和编辑邮件，支持 POP3 和 IMAP 通信协议，检查用户的设定是否正确。系统界面简明，操作简单。

本项目是基于 Android 手机平台的邮件系统，使 Android 手机拥有个性的邮件系统，使手机显得更方便和智能，与人们更为接近，让手机主人可以随时随地处理工作事务或者是与朋友联系，使人们的生活更加

多样化，也使设计者更加熟练掌握 Android 技术。

3.2 系统需求分析

知识点讲解：光盘:视频\视频讲解\第 3 章\系统需求分析.avi

根据项目的目标，可分析出系统的基本需求，下面从软件设计的角度来描述系统的功能，并且使用例图描述邮件系统的功能模块，大致分成 5 部分来概括，即邮箱类型设置、邮箱收取设置、邮箱发送设置、邮箱用户检查和用户邮件编辑。

3.2.1 构成模块

本系统的构成模块如图 3-1 所示。各个模块的具体说明如下所示。

1. 邮箱类型设置

此模块的功能是设置通信协议。

（1）POP3 协议

- ☑ 目标：使得用户可以收发邮件到本地。
- ☑ 前置条件：成功登录邮件系统。
- ☑ 基本事件流：
  - ➢ 用户单击 next 按钮。
  - ➢ 程序进入邮箱收取设置。

（2）IMAP 协议

- ☑ 目标：使得用户可以在线收发邮件。
- ☑ 前置条件：成功登录邮件系统。
- ☑ 基本事件流：
  - ➢ 用户单击 next 按钮。
  - ➢ 程序进入邮箱收取设置。

邮箱类型设置界面结构如图 3-2 所示。

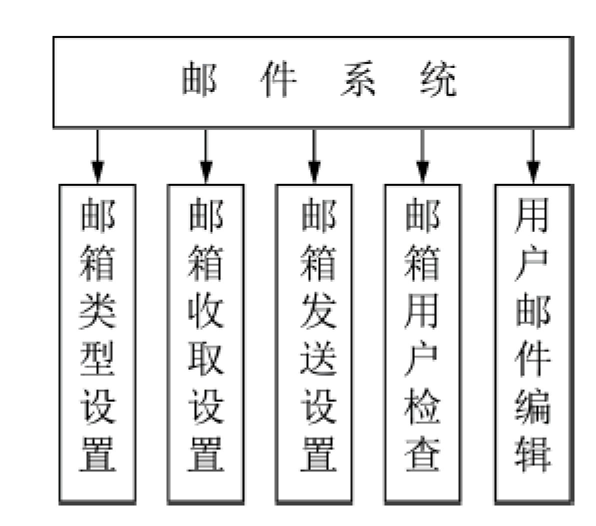


图 3-1　系统构成模块

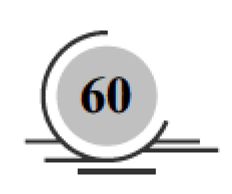
通信协议
POP3
IMAP

图 3-2　邮箱类型设置界面结构

2. 邮箱收取设置

当用户选定通信协议后，可以进行邮箱收取设置。

- ☑ 目标：设定用户基本信息。



☑　前置条件：程序运行在用户基本信息设定界面。

☑　基本事件流：

- ➢　用户填写用户名和密码。
- ➢　用户填写服务器名和端口。
- ➢　用户填写加密协议。
- ➢　用户设定邮件删除期限。
- ➢　用户单击 next 按钮。

邮箱收取设置界面结构如图 3-3 所示。

3. 邮箱发送设置

本模块用于设置邮箱发送。

☑　目标：设定邮箱发送。

☑　前置条件：程序运行在邮箱发送设定界面。

☑　基本事件流：

- ➢　用户填写服务器名和端口。
- ➢　用户单击 next 按钮。

邮箱发送设置界面结构如图 3-4 所示。

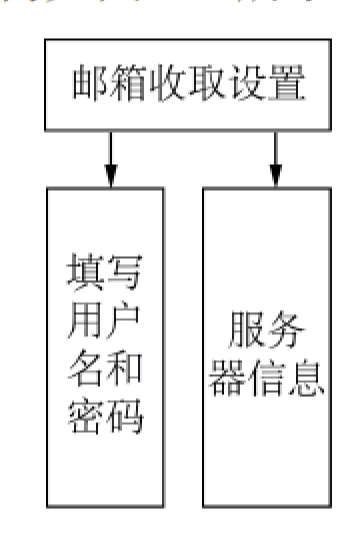


图 3-3　邮箱收取设置界面结构

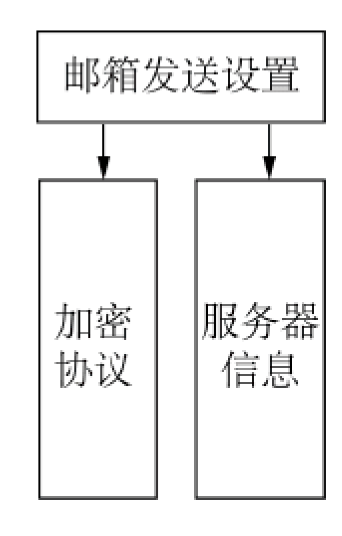


图 3-4　邮箱发送设置界面结构

4. 邮箱用户检查

此模块的功能是邮箱用户检查。

（1）用户名和密码验证

☑　目标：验证用户名和密码正确性。

☑　前置条件：程序运行主界面。

（2）接收地址验证

☑　目标：验证接收地址正确性。

☑　前置条件：程序运行目录界面。

（3）发送地址验证

☑　目标：验证发送地址正确性。

☑　前置条件：程序运行目录界面。

☑　基本事件流：

- ➢　用户单击 next 按钮。

邮箱用户检查界面结构如图 3-5 所示。

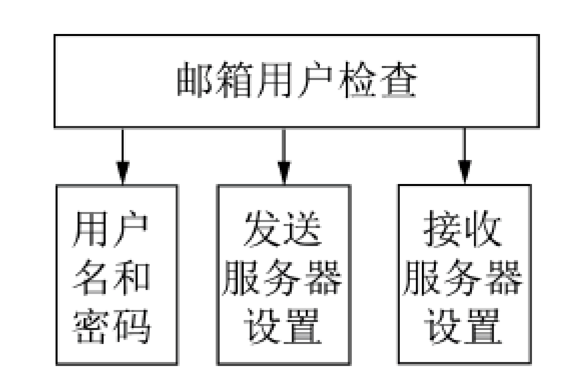


图 3-5　邮箱用户检查界面结构

5. 用户邮件编辑

此模块的功能是实现邮件编辑。

☑ 目标：编辑邮件。

☑ 前置条件：进入邮件编辑界面。

☑ 基本事件流：

- ➢ 用户填写收件人地址。
- ➢ 用户填写标题。
- ➢ 用户填写邮件内容。
- ➢ 用户单击 send 按钮。

用户邮件编辑界面结构如图 3-6 所示。

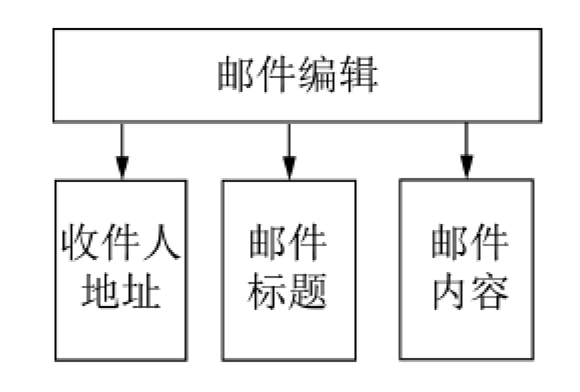


图 3-6　用户邮件编辑界面结构

3.2.2　系统流程

邮件系统流程图如图 3-7 所示。

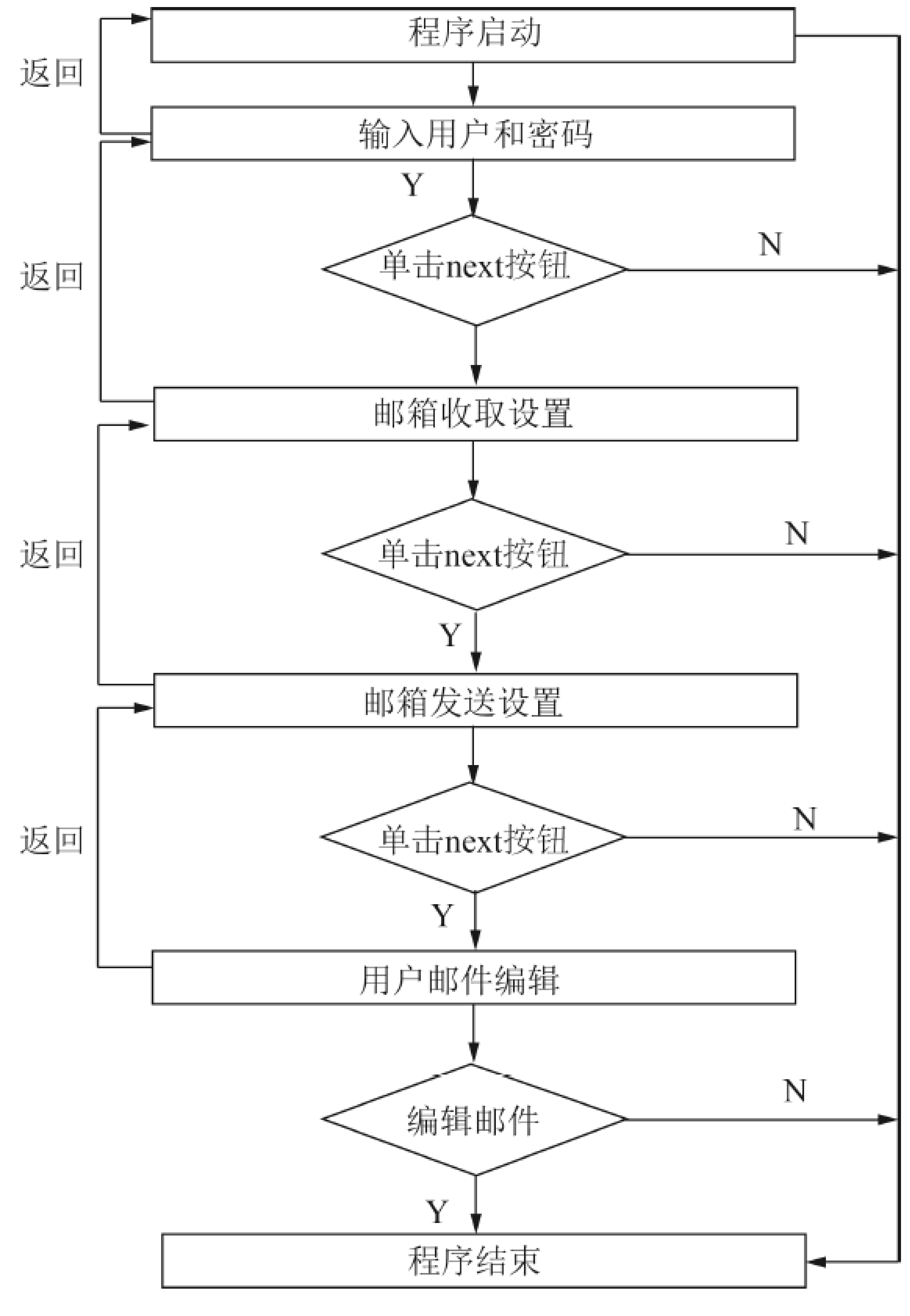


图 3-7　邮件系统流程图

3.2.3　功能结构图

本章邮件系统的完整功能结构如图 3-8 所示。

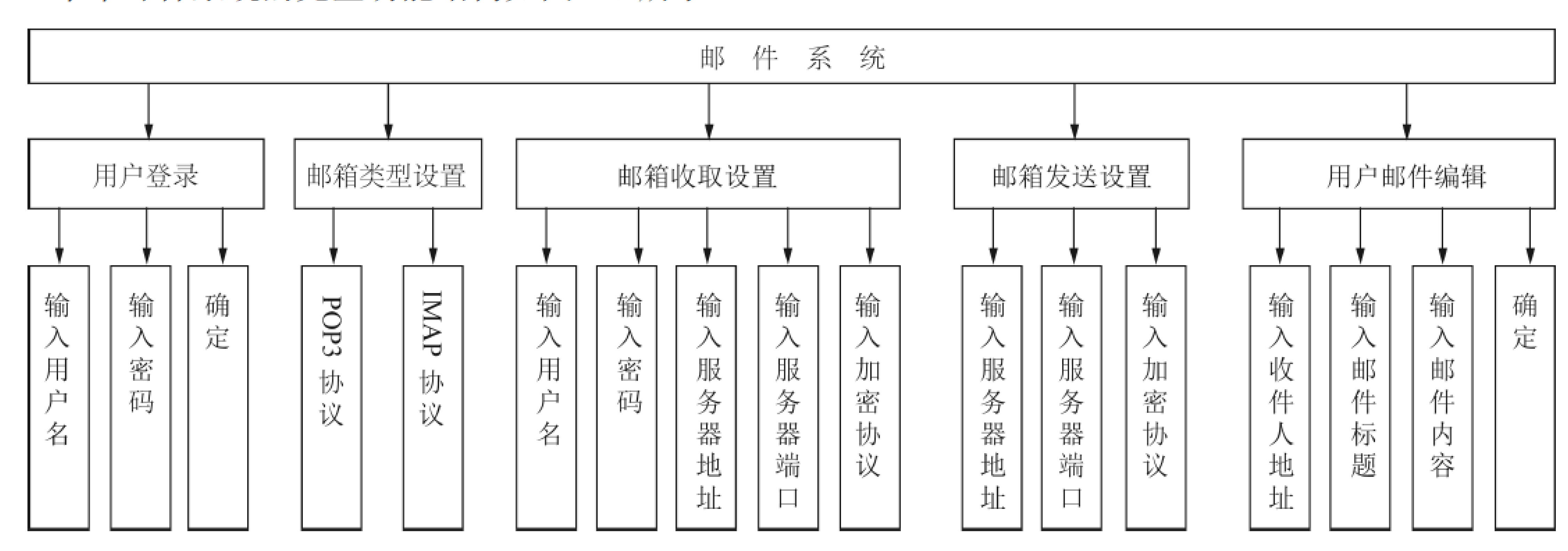


图 3-8　完整功能结构图

3.2.4　系统功能说明

本章邮件系统各个模块功能的说明如表 3-1 所示。

表 3-1　模块结构功能说明信息表

功 能 类 别	子　功　能	功 能 类 别	子　功　能
用户登录	输入用户名	邮箱收取设置	输入加密协议
	输入密码	邮箱发送设置	输入服务器地址
	进入邮件收取设置		输入服务器端口
邮箱类型设置	POP3 协议		输入加密协议
	IMAP 协议	用户邮件编辑	输入收件人地址
邮箱收取设置	输入用户名		输入邮件标题
	输入密码		输入邮件内容
	输入服务器地址		单击发送按钮
	输入服务器端口		

3.2.5　系统需求

（1）系统性能需求

根据 Android 手机系统要求无响应时间为 5 秒，所以就有如下性能要求：

☑　邮箱类型设置，程序响应时间最长不能超过 5 秒。

☑　邮箱收取设置，程序响应时间最长不能超过 5 秒。

☑　邮箱发送设置，程序响应时间最长不能超过 5 秒。

☑　邮箱用户检查，程序响应时间最长不能超过 5 秒。

☑　用户邮件编辑，程序响应时间最长不能超过 5 秒。

（2）运行环境需求

☑ 操作系统：Android 手机基于 Linux 操作系统。
☑ 支持环境：Android 1.5 - 2.0.1 版本。
☑ 开发环境：Eclipse 3.5 ADT 0.95。

3.3 数据存储设计

知识点讲解：光盘:视频\视频讲解\第 3 章\数据存储设计.avi

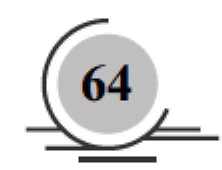
基于 Windows 或者是 Linux 的大型系统开发，数据库使用的都是专业数据库系统，Android 开发平台提供了几种数据存储：Android 系统中自带 iSQLite 数据库，数据接口共享数据（SharedPreferences）模式保存数据、文件方式保存数据、内容提供器（ContextProvider）和网络方式保存数据。数据库是存放数据的仓库，只不过这个仓库是在计算机存储设备上，而且数据是按一定格式存放的。本实例采用 SharedPreferences 保存数据。

SharedPreferences 以 XML 格式自动保存，在 DDMS 中的 File ExPlore 中展开到/data/data/<package name>/shared_prefs 目录下，生成 AndroidMail.Main.xml 文件。

3.3.1 用户信息类

定义用户信息 Account.java 类，此类将保存系统用户有关的所有信息，为 SharedPreferences 模式保存数据提供用户实例对象，对应的部分代码如下所示。

```
public class Account implements Serializable {
    public static final int DELETE_POLICY_NEVER = 0;
    public static final int DELETE_POLICY_7DAYS = 1;
    public static final int DELETE_POLICY_ON_DELETE = 2;
    private static final long serialVersionUID = 2975156672298625121L;
    String mUuid;                                   //邮件用户 ID
    String mStoreUri;                               //邮件源地址
    String mLocalStoreUri;
    String mSenderUri;                              //邮件目的地址
    String mDescription;                            //邮件内容
    String mName;                                   //用户名
    String mEmail;
    int mAutomaticCheckIntervalMinutes;
    long mLastAutomaticCheckTime;
    boolean mNotifyNewMail;
    String mDraftsFolderName;
    String mSentFolderName;
    String mTrashFolderName;
    String mOutboxFolderName;
    int mAccountNumber;
    boolean mVibrate;
    String mRingtoneUri;
    int mDeletePolicy;
      //初始化
    public Account(Context context) {
```



```
        mUuid = UUID.randomUUID().toString();
        mLocalStoreUri = "local://localhost/" + context.getDatabasePath(mUuid + ".db");
        mAutomaticCheckIntervalMinutes = -1;
        mAccountNumber = -1;
        mNotifyNewMail = true;
        mVibrate = false;
        mRingtoneUri = "content://settings/system/notification_sound";
    }
  //刷新指定用户
    Account(Preferences preferences, String uuid) {
        this.mUuid = uuid;
        refresh(preferences);
    }
      //刷新
    public void refresh(Preferences preferences) {
        mStoreUri = Utility.base64Decode(preferences.mSharedPreferences.getString(mUuid
                + ".storeUri", null));
```

☑ 通过 SharedPreferences 对象的 getString ()方法获取保存在其中的值，对应部分代码如下所示。

```
        mLocalStoreUri = preferences.mSharedPreferences.getString(mUuid + ".localStoreUri", null);
        String senderText = preferences.mSharedPreferences.getString(mUuid + ".senderUri", null);
        if (senderText == null) {
    //获取 ID
            senderText = preferences.mSharedPreferences.getString(mUuid + ".transportUri", null);
        }
     //转换编码方式
        mSenderUri = Utility.base64Decode(senderText);
        mDescription = preferences.mSharedPreferences.getString(mUuid + ".description", null);
     //获取与此用户身份有关的信息
        mName = preferences.mSharedPreferences.getString(mUuid + ".name", mName);
        mEmail = preferences.mSharedPreferences.getString(mUuid + ".email", mEmail);
        mAutomaticCheckIntervalMinutes = preferences.mSharedPreferences.getInt(mUuid
                + ".automaticCheckIntervalMinutes", -1);
        mLastAutomaticCheckTime = preferences.mSharedPreferences.getLong(mUuid
                + ".lastAutomaticCheckTime", 0);
        mNotifyNewMail = preferences.mSharedPreferences.getBoolean(mUuid + ".notifyNewMail",
                false);
        mDraftsFolderName = preferences.mSharedPreferences.getString(mUuid + ".draftsFolderName",
                "Drafts");
        mSentFolderName = preferences.mSharedPreferences.getString(mUuid + ".sentFolderName",
                "Sent");
        mTrashFolderName = preferences.mSharedPreferences.getString(mUuid + ".trashFolderName",
                "Trash");
        mOutboxFolderName = preferences.mSharedPreferences.getString(mUuid + ".outboxFolderName",
                "Outbox");
        mAccountNumber = preferences.mSharedPreferences.getInt(mUuid + ".accountNumber", 0);
        mVibrate = preferences.mSharedPreferences.getBoolean(mUuid + ".vibrate", false);
        mRingtoneUri = preferences.mSharedPreferences.getString(mUuid + ".ringtone",
                "content://settings/system/notification_sound");
    }
    public String getUuid() {
        return mUuid;
```

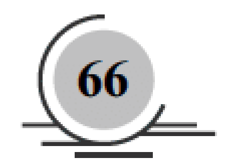
```
    }
    public String getStoreUri() {
        return mStoreUri;
    }
```

☑ 通过各属性的 set ()方法赋值，对应部分代码如下所示。

```
//为属性赋值
public void setStoreUri(String storeUri) {
        this.mStoreUri = storeUri;
    }
    public String getSenderUri() {
        return mSenderUri;
    }
    public void setSenderUri(String senderUri) {
        this.mSenderUri = senderUri;
    }
    public String getDescription() {
        return mDescription;
    }
    public void setDescription(String description) {
        this.mDescription = description;
    }
    public String getName() {
        return mName;
    }
    public void setName(String name) {
        this.mName = name;
    }
    public String getEmail() {
        return mEmail;
    }
    public void setEmail(String email) {
        this.mEmail = email;
    }
    public boolean isVibrate() {
        return mVibrate;
    }
    public void setVibrate(boolean vibrate) {
        mVibrate = vibrate;
    }
    public String getRingtone() {
        return mRingtoneUri;
    }
public void setRingtone(String ringtoneUri) {
        mRingtoneUri = ringtoneUri;
    }
```

☑ 定义 delete()方法删除指定的 account 实例，对象 SharedPreferences.Editor 中的 Remove()方法执行删除值操作。commit()方法对所做的修改进行提交，对应部分代码如下所示。

```
    public void delete(Preferences preferences) {
        String[ ] uuids = preferences.mSharedPreferences.getString("accountUuids", "").split(",");
        StringBuffer sb = new StringBuffer();
```



```
        for (int i = 0, length = uuids.length; i < length; i++) {
            if (!uuids[i].equals(mUuid)) {
                if (sb.length() > 0) {
                    sb.append(',');
                }
                sb.append(uuids[i]);
            }
        }
        String accountUuids = sb.toString();
       //定义 SharedPreferences.Editor 对象，清除指定值
        SharedPreferences.Editor editor = preferences.mSharedPreferences.edit();
        editor.putString("accountUuids", accountUuids);
        editor.remove(mUuid + ".storeUri");
        editor.remove(mUuid + ".localStoreUri");
        editor.remove(mUuid + ".senderUri");
        editor.remove(mUuid + ".description");
        editor.remove(mUuid + ".name");
        editor.remove(mUuid + ".email");
        editor.remove(mUuid + ".automaticCheckIntervalMinutes");
        editor.remove(mUuid + ".lastAutomaticCheckTime");
        editor.remove(mUuid + ".notifyNewMail");
        editor.remove(mUuid + ".deletePolicy");
        editor.remove(mUuid + ".draftsFolderName");
        editor.remove(mUuid + ".sentFolderName");
        editor.remove(mUuid + ".trashFolderName");
        editor.remove(mUuid + ".outboxFolderName");
        editor.remove(mUuid + ".accountNumber");
        editor.remove(mUuid + ".vibrate");
        editor.remove(mUuid + ".ringtone");
        editor.remove(mUuid + ".transportUri");
        //提交所做的操作
        editor.commit();
    }
```

☑ 定义 save()方法保存指定的 account 实例，对象 SharedPreferences.Editor 中的方法 putString()保存指定的值，该方法第一个参数是键名；第二个参数是值，对应部分代码如下所示。

```
    public void save(Preferences preferences) {
        if (!preferences.mSharedPreferences.getString("accountUuids", "").contains(mUuid)) {
            Account[ ] accounts = preferences.getAccounts();
            int[ ] accountNumbers = new int[accounts.length];
            for (int i = 0; i < accounts.length; i++) {
                accountNumbers[i] = accounts[i].getAccountNumber();
            }
            Arrays.sort(accountNumbers);
            for (int accountNumber : accountNumbers) {
                if (accountNumber > mAccountNumber + 1) {
                    break;
                }
                mAccountNumber = accountNumber;
            }
            mAccountNumber++;
```

```
                String accountUuids = preferences.mSharedPreferences.getString("accountUuids", "");
                accountUuids += (accountUuids.length() != 0 ? "," : "") + mUuid;
                SharedPreferences.Editor editor = preferences.mSharedPreferences.edit();
            //保存 accountUuids 名的值
                editor.putString("accountUuids", accountUuids);
            //提交所做的操作
                editor.commit();
        }
        SharedPreferences.Editor editor = preferences.mSharedPreferences.edit();
        editor.putString(mUuid + ".storeUri", Utility.base64Encode(mStoreUri));
        editor.putString(mUuid + ".localStoreUri", mLocalStoreUri);
        editor.putString(mUuid + ".senderUri", Utility.base64Encode(mSenderUri));
        editor.putString(mUuid + ".description", mDescription);
        editor.putString(mUuid + ".name", mName);
        editor.putString(mUuid + ".email", mEmail);
        editor.putInt(mUuid + ".automaticCheckIntervalMinutes", mAutomaticCheckIntervalMinutes);
        editor.putLong(mUuid + ".lastAutomaticCheckTime", mLastAutomaticCheckTime);
        editor.putBoolean(mUuid + ".notifyNewMail", mNotifyNewMail);
        editor.putInt(mUuid + ".deletePolicy", mDeletePolicy);
        editor.putString(mUuid + ".draftsFolderName", mDraftsFolderName);
        editor.putString(mUuid + ".sentFolderName", mSentFolderName);
        editor.putString(mUuid + ".trashFolderName", mTrashFolderName);
        editor.putString(mUuid + ".outboxFolderName", mOutboxFolderName);
        //保存整型数据
        editor.putInt(mUuid + ".accountNumber", mAccountNumber);
        //保存逻辑型数据
        editor.putBoolean(mUuid + ".vibrate", mVibrate);
        editor.putString(mUuid + ".ringtone", mRingtoneUri);
        editor.remove(mUuid + ".transportUri");
        editor.commit();
    }
}
```

3.3.2 SharedPreferences

定义 Preferences.java 类，该类基于 SharedPreferences 类。使用 getSharedPreferences()方法返回 mShared-Preferences 对象，对应部分代码如下所示。

```
public class Preferences {
    private static Preferences preferences;
    SharedPreferences mSharedPreferences;
    private Preferences(Context context) {
    //读取数据
        mSharedPreferences = context.getSharedPreferences("AndroidMail.Main", Context.MODE_PRIVATE);
    }
    public static synchronized Preferences getPreferences(Context context) {
        if (preferences == null) {
            preferences = new Preferences(context);
        }
        return preferences;
    }
```

☑ 定义 getAccounts()方法，返回 accountUuids 对应的值，对应部分代码如下所示。

```
public Account[ ] getAccounts() {
    String accountUuids = mSharedPreferences.getString("accountUuids", null);
    if (accountUuids == null || accountUuids.length() == 0) {
        return new Account[ ] { };
    }
    String[ ] uuids = accountUuids.split(",");
    Account[ ] accounts = new Account[uuids.length];
    for (int i = 0, length = uuids.length; i < length; i++) {
        accounts[i] = new Account(this, uuids[i]);
    }
    return accounts;
}
```

☑ 定义 getAccountByContentUri()方法，返回指定邮箱类型对应的 URL 值，对应部分代码如下所示。

```
public Account getAccountByContentUri(Uri uri) {
    if (!"content".equals(uri.getScheme()) || !"accounts".equals(uri.getAuthority())) {
        return null;
    }
    String uuid = uri.getPath().substring(1);
    if (uuid == null) {
        return null;
    }
    String accountUuids = mSharedPreferences.getString("accountUuids", null);
    if (accountUuids == null || accountUuids.length() == 0) {
        return null;
    }
    String[ ] uuids = accountUuids.split(",");
    for (int i = 0, length = uuids.length; i < length; i++) {
        if (uuid.equals(uuids[i])) {
            return new Account(this, uuid);
        }
    }
    return null;
}
```

☑ 定义 getDefaultAccount()方法，返回默认的用户 ID，对应部分代码如下所示。

```
public Account getDefaultAccount() {
    String defaultAccountUuid = mSharedPreferences.getString("defaultAccountUuid", null);
    Account defaultAccount = null;
    Account[ ] accounts = getAccounts();
    if (defaultAccountUuid != null) {
        for (Account account : accounts) {
            if (account.getUuid().equals(defaultAccountUuid)) {
                defaultAccount = account;
                break;
            }
        }
    }
    if (defaultAccount == null) {
        if (accounts.length > 0) {
            defaultAccount = accounts[0];
```

```
                setDefaultAccount(defaultAccount);
            }
        }
        return defaultAccount;
    }
    public void setDefaultAccount(Account account) {
        mSharedPreferences.edit().putString("defaultAccountUuid", account.getUuid()).commit();
    }
```

☑ 定义 setEnableDebugLogging()方法赋值是否开启调试信息，geteEnableDebugLogging()读取调试信息开启情况，对应部分代码如下所示。

```
    public void setEnableDebugLogging(boolean value) {
        mSharedPreferences.edit().putBoolean("enableDebugLogging", value).commit();
    }
    public boolean geteEnableDebugLogging() {
        return mSharedPreferences.getBoolean("enableDebugLogging", false);
    }
  public void setEnableSensitiveLogging(boolean value) {
   //直接修改 enableSensitiveLogging 的值
        mSharedPreferences.edit().putBoolean("enableSensitiveLogging", value).commit();
    }
    public boolean getEnableSensitiveLogging() {
        return mSharedPreferences.getBoolean("enableSensitiveLogging", false);
    }
    public void save() {
    }
    public void clear() {
          //清除对象中的键名
        mSharedPreferences.edit().clear().commit();
    }
    public void dump() {
        if (Config.LOGV) {
            for (String key : mSharedPreferences.getAll().keySet()) {
                Log.v(Email.LOG_TAG, key + " = " + mSharedPreferences.getAll().get(key));
            }
        }
    }
```

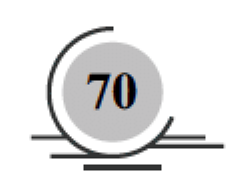
3.4 具体编码

知识点讲解：光盘:视频\视频讲解\第 3 章\具体编码.avi

经过前面的讲解，制作本邮件系统实例项目的前期工作已经结束。在接下来的内容中，将详细讲解本项目的具体编码过程。

3.4.1 欢迎界面

欢迎界面是整个项目的入口，通过入口可进入到系统的其他功能，欢迎界面如图 3-9 所示。



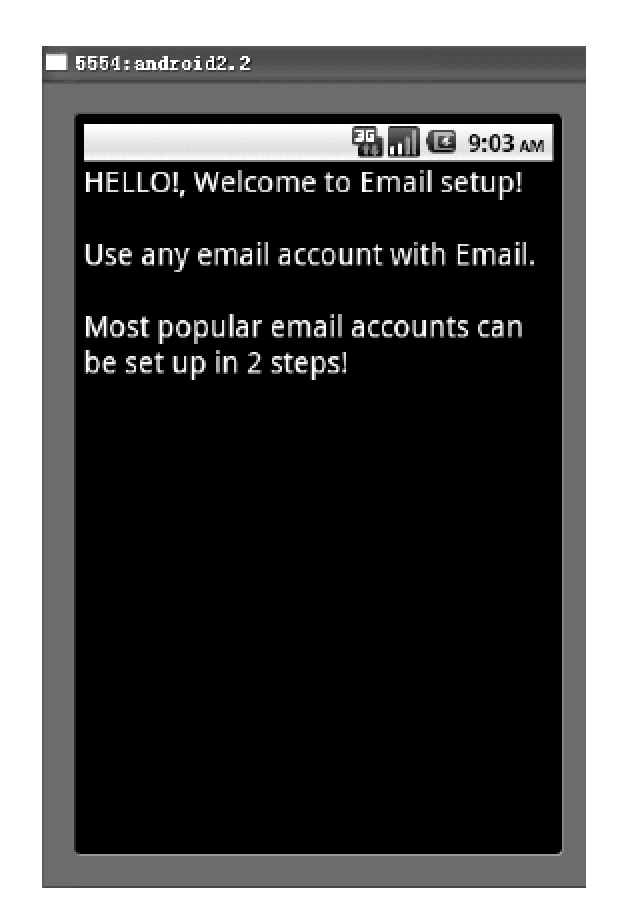


图 3-9　欢迎界面

（1）编写文件 WelActivity.java，在此定义项目的欢迎界面，主要代码如下所示。

☑ 定义 WelActivity 类继承 ListActivity 类，ListActivity 类又继承 Activity。ListActivity 默认绑定了一个 ListView（列表视图）界面组件，提供一些与视图、处理相关的操作，部分代码如下。

```
public class WelActivity extends ListActivity implements OnItemClickListener, OnClickListener{
    private static final String EXTRA_ACCOUNT = "account";
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //加载 activity_wel.xml 布局
        setContentView(R.layout.activity_wel);
        ListView listView = getListView();
        listView.setOnItemClickListener(this);
        listView.setItemsCanFocus(false);
        //获取指定的组件
        listView.setEmptyView(findViewById(R.id.empty));
        findViewById(R.id.add_new_account).setOnClickListener(this);
    }
    public void onItemClick(AdapterView<?> parent, View arg1, int position, long arg3) {
        Account account = (Account)parent.getItemAtPosition(position);
        Intent intent = new Intent(this, EmailCpsActivity.class);
        //启动前传值在 Activity 中
        intent.putExtra(EXTRA_ACCOUNT , account);
        //启动 Activity
        startActivity(intent);
    }
```

☑ 定义 onResume()方法，该方法在窗口暂停后回调。所有窗体都继承 Activity 类，因此在窗体设计类中都应该包含此类方法，另外与窗体调用有关的方法有 onstart()方法、onCreate()方法、onpause()方法、onstop()方法、onrestart()方法和 ondestroy()方法。

```
    public void onClick(View v) {
        if (v.getId() == R.id.add_new_account) {
            Intent intent = new Intent(this, AccountSetupActivity.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);}
    }
```

```
        //暂停后调用
         public void onResume() {
              super.onResume();
              refresh();
         }
         //刷新操作
         private void refresh() {
              Account[ ] accounts = Preferences.getPreferences(this).getAccounts();
              getListView().setAdapter(new AccountsAdapter(accounts));
         }
 @Override
     class AccountsAdapter extends ArrayAdapter<Account> {
         public AccountsAdapter(Account[ ] accounts) {
              super(WelActivity.this, 0, accounts);
         }
         public View getView(int position, View convertView, ViewGroup parent) {
              Account account = getItem(position);
              View view;
              if (convertView != null) {
                   view = convertView;
              }else {
             view = getLayoutInflater().inflate(R.layout.accounts_item, parent, false);
              }
              AccountViewHolder holder = (AccountViewHolder) view.getTag();
              if (holder == null) {
                   holder = new AccountViewHolder();
                   holder.description = (TextView) view.findViewById(R.id.description);
                   holder.email = (TextView) view.findViewById(R.id.email);
                   view.setTag(holder);
              }
              holder.description.setText(account.getDescription());
              holder.email.setText(account.getEmail());
              if (account.getEmail().equals(account.getDescription())) {
                   holder.email.setVisibility(View.GONE);
              }
              return view;
         }
         class AccountViewHolder {
              public TextView description;
              public TextView email;
         }
    }
```

（2）Android 开发是可视化界面开发，每个窗口都有唯一的布局 XML 配置文件，窗口的各种布局效果都有对应的标签表示，例如，图像、文字和控件位置的设置等，程序在运行时读取配置文件，满足不同的界面应用。

☑ 本实例主界面的布局文件是 AndroidManifest.xml，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
>
<ListView
        android:id="@android:id/list"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1.0"
        />
<LinearLayout
        android:id="@+id/empty"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical">
<TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:text="@string/accounts_welcome"
            android:textColor="?android:attr/textColorPrimary" />
<View
            android:layout_width="fill_parent"
            android:layout_height="0px"
            android:layout_weight="1" />
</LinearLayout>
<RelativeLayout
            android:layout_width="fill_parent"
            android:layout_height="54dip"
            android:background="@android:drawable/menu_full_frame">
<Button
                android:id="@+id/add_new_account"
                android:layout_width="wrap_content"
                android:minWidth="100dip"
                android:layout_height="wrap_content"
                android:text="@string/next_action"
                android:drawableRight="@drawable/button_indicator_next"
                android:layout_alignParentRight="true"
                android:layout_centerVertical="true" />
</RelativeLayout>
</LinearLayout>
```

☑ 以上代码中采用 LinearLayout 布局，android:orientation="vertical"实现控件水平方向排列，android:orientation="horizontal"实现控件竖直方向排列。标签 RelativeLayout 布局提供一个容器，所有控件在容器中的位置按相对位置来计算。

☑ Android:id 定义控件的 ID，程序根据 ID 可以访问相应的控件。代码中定义了 list、empty 和 add_new_account，分别表示 ListView 控件、LinearLayout 控件和 Button 控件。

android:layout_width="fill_parent"和 android:layout_height="wrap_content"表示控制宽度占全屏，控制高度适应容器的大小。总之，fill_parent 就是让控件宽或者高占全屏，而 wrap_content 是让控件的高或宽仅仅把控件里的内容包裹住，而不是全屏。

3.4.2 系统主界面

系统根据输入的用户名和密码设置用户的属性，如图 3-10 所示。

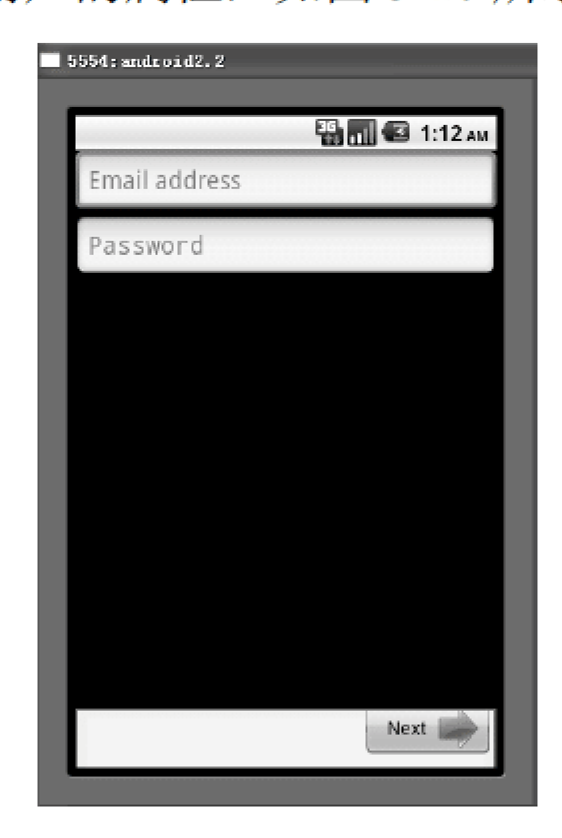


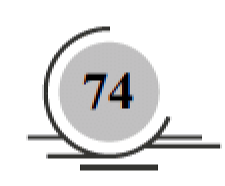
图 3-10 用户属性设置

（1）编写文件 AccountSetupActivity.java，在此定义用户设置界面。

☑ 定义 onCreate()方法初始化窗体，方法参数 savedInstanceState 保存当前 Activity 的状态信息。定义监听器 setOnClickListener()、addTextChangedListener 和 addTextChangedListener()，对应部分代码如下所示。

```
public class AccountSetupActivity extends Activity implements OnClickListener,TextWatcher{
     private final static int DIALOG_NOTE = 1;
     private EmailAddressValidator mEmailValidator = new EmailAddressValidator();
    private EditText mEmailView;
    private EditText mPasswordView;
    private Button mNextButton;
    private Account mAccount;
    private Provider mProvider;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //加载 activity_account_setup.xml 布局文件
        setContentView(R.layout.activity_account_setup);
        mEmailView = (EditText)findViewById(R.id.account_email);
        mPasswordView = (EditText)findViewById(R.id.account_password);
        mNextButton = (Button)findViewById(R.id.next);
        //定义监听器
        mNextButton.setOnClickListener(this);
        mEmailView.addTextChangedListener(this);
        mPasswordView.addTextChangedListener(this);
    }
```

☑ 定义 onCreateDialog()方法创建一个对话框，方法参数表示对话框 ID。AlertDialog.Builder(this)创建一个 AlertDialog 对话框，setIcon()方法设置对话框图片，setTitle()方法设置显示标题，setMessage()方法定义提示信息内容。setPositiveButton()方法设置确定按钮的一些属性，第一个参数为按钮上显示内容；第二个参数为 DialogInterface.OnClickListener()监听器对象，监听单击事件。对应部分代码如下所示。



```
//创建一个对话框窗口
public Dialog onCreateDialog(int id) {
    if (id == DIALOG_NOTE) {
        if (mProvider != null && mProvider.note != null) {
            return new AlertDialog.Builder(this)
                .setIcon(android.R.drawable.ic_dialog_alert)
                .setTitle(android.R.string.dialog_alert_title)
                .setMessage(mProvider.note)
                .setPositiveButton(
                        getString(R.string.okay_action),
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int which) {
                                finishAutoSetup();
                            }
                        })
                .setNegativeButton(
                        getString(R.string.cancel_action),
                        null)
                .create();
        }
    }
    return null;
}
```

☑ 当用户单击确定按钮后调用 finishAutoSetup()方法，实例化 URI 对象保存用户邮件的详细信息，例如，用户名、密码、主机地址和端口。最终封装在 Account 类的实例 mAccount，调用对应的 set 方法赋值。邮件用户设置不正确时调用 onManualSetup()方法，若用户设置正确则调用 actionCheck-Settings()方法。对应部分代码如下所示。

```
 private void finishAutoSetup() {
    String email = mEmailView.getText().toString().trim();
    String password = mPasswordView.getText().toString().trim();
    String[ ] emailParts = email.split("@");
    String user = emailParts[0];
    String domain = emailParts[1];
    URI incomingUri = null;
    URI outgoingUri = null;
    try {
        String incomingUsername = mProvider.incomingUsernameTemplate;
        incomingUsername = incomingUsername.replaceAll("\\$email", email);
        incomingUsername = incomingUsername.replaceAll("\\$user", user);
        incomingUsername = incomingUsername.replaceAll("\\$domain", domain);
        URI incomingUriTemplate = mProvider.incomingUriTemplate;
        incomingUri = new URI(incomingUriTemplate.getScheme(), incomingUsername + ":" + password,
incomingUriTemplate.getHost(), incomingUriTemplate.getPort(), null,
                null, null);
        String outgoingUsername = mProvider.outgoingUsernameTemplate;
        outgoingUsername = outgoingUsername.replaceAll("\\$email", email);
        outgoingUsername = outgoingUsername.replaceAll("\\$user", user);
        outgoingUsername = outgoingUsername.replaceAll("\\$domain", domain);
        URI outgoingUriTemplate = mProvider.outgoingUriTemplate;
        outgoingUri = new URI(outgoingUriTemplate.getScheme(), outgoingUsername + ":" + password,
```

```
outgoingUriTemplate.getHost(), outgoingUriTemplate.getPort(), null,
                                null, null);
            } catch (URISyntaxException use) {
                onManualSetup();
                return;
            }
            //给 Account 对象的属性赋值
            mAccount = new Account(this);
            mAccount.setName(getOwnerName());
            mAccount.setEmail(email);
            mAccount.setStoreUri(incomingUri.toString());
            mAccount.setSenderUri(outgoingUri.toString());
            mAccount.setDraftsFolderName(getString(R.string.special_mailbox_name_drafts));
            mAccount.setTrashFolderName(getString(R.string.special_mailbox_name_trash));
            mAccount.setOutboxFolderName(getString(R.string.special_mailbox_name_outbox));
            mAccount.setSentFolderName(getString(R.string.special_mailbox_name_sent));
            if (incomingUri.toString().startsWith("imap")) {
                mAccount.setDeletePolicy(Account.DELETE_POLICY_ON_DELETE);
            }
            AccountCheckSettings.actionCheckSettings(this, mAccount, true, true);
        }
```

☑ 定义 getOwnerName()方法获取当前用户，通过共享数据接口取得用户 account 对象。getName()返回具体的用户名。单击向下按钮调用 findProviderForDomain()方法，从 providers_product.xml 配置文件中读取已有账户信息。对应部分代码如下所示。

```
     private String getOwnerName() {
            String name = null;
            //通过 SharedPreferences 对象取得用户名
            Account account = Preferences.getPreferences(this).getDefaultAccount();
            if (account != null) {
                name = account.getName();
            }
            return name;
        }
     private void onNext() {
            String email = mEmailView.getText().toString().trim();
            String[ ] emailParts = email.split("@");
            String domain = emailParts[1].trim();
            mProvider = findProviderForDomain(domain);
            if (mProvider == null) {
            //默认设置用户调用 manual
                onManualSetup();
                return;
            }
            if (mProvider.note != null) {
              //显示对话框
                showDialog(DIALOG_NOTE);
            }
            else {
                finishAutoSetup();
            }
```

```
    }
    private Provider findProviderForDomain(String domain) {
        Provider p = findProviderForDomain(domain, R.xml.providers_product);
        if (p == null) {
            p = findProviderForDomain(domain, R.xml.providers);
        }
        return p;
    }
```

☑ 如果 providers_product 文件中没有用户信息，通过 findProviderForDomain()方法读取 providers 文件中提供接收邮件和发送邮件的服务器的信息，最后将 id、lable、domain、uri 保存在 provider 实例中，对应部分代码如下所示。

```
   private String getXmlAttribute(XmlResourceParser xml, String name) {
        int resId = xml.getAttributeResourceValue(null, name, 0);
        if (resId == 0) {
            return xml.getAttributeValue(null, name);
        }
        else {
            return getString(resId);
        }
    }
        //读取资源文件 xml
    private Provider findProviderForDomain(String domain, int resourceId) {
        try {
            XmlResourceParser xml = getResources().getXml(resourceId);
            int xmlEventType;
            Provider provider = null;
              //逐行读取 XML 文件
            while ((xmlEventType = xml.next()) != XmlResourceParser.END_DOCUMENT) {
                if (xmlEventType == XmlResourceParser.START_TAG
                            && "provider".equals(xml.getName())
                            && domain.equalsIgnoreCase(getXmlAttribute(xml, "domain"))) {
                    provider = new Provider();
                    //读取指定键值的值
                    provider.id = getXmlAttribute(xml, "id");
                    provider.label = getXmlAttribute(xml, "label");
                    provider.domain = getXmlAttribute(xml, "domain");
                    provider.note = getXmlAttribute(xml, "note");
                }
                else if (xmlEventType == XmlResourceParser.START_TAG
                    && "incoming".equals(xml.getName())
                    && provider != null) {
                        provider.incomingUriTemplate = new URI(getXmlAttribute(xml, "uri"));
                        provider.incomingUsernameTemplate = getXmlAttribute(xml, "username");
                }
                else if (xmlEventType == XmlResourceParser.START_TAG
                    && "outgoing".equals(xml.getName())
                    && provider != null) {
                        provider.outgoingUriTemplate = new URI(getXmlAttribute(xml, "uri"));
                        provider.outgoingUsernameTemplate = getXmlAttribute(xml, "username");
                    }
```

```
                    else if (xmlEventType == XmlResourceParser.END_TAG
                            && "provider".equals(xml.getName())
                            && provider != null) {
                        return provider;
                    }
                }
            }
        }
        catch (Exception e) {
            Log.e(Email.LOG_TAG, "Error while trying to load provider settings.", e);
        }
        return null;
    }
```

☑ 定义 onManualSetup()方法重新设置用户名和密码，将新设定的信息封装在 URI 实例中。若设定失败，makeText()方法在主界面前显示出错内容。用户名和密码设定后进入 AccountSetupAccountType 对象的 actionSelectAccountType()方法。对应部分代码如下所示。

```
    private void onManualSetup() {
        String email = mEmailView.getText().toString().trim();
        String password = mPasswordView.getText().toString().trim();
        String[ ] emailParts = email.split("@");
        String user = emailParts[0].trim();
        String domain = emailParts[1].trim();
        mAccount = new Account(this);
        mAccount.setName(getOwnerName());
        mAccount.setEmail(email);
        try {//实例化 URI，为 uri 赋值
            URI uri = new URI("placeholder", user + ":" + password, domain, -1, null, null, null);
            mAccount.setStoreUri(uri.toString());
            mAccount.setSenderUri(uri.toString());
        } catch (URISyntaxException use) {
        //URL 地址出错将给出提示
            Toast.makeText(this,R.string.account_setup_username_password_toast,
Toast.LENGTH_LONG).show();
            mAccount = null;
            return;
        } //为 Account 对象的属性赋值
        mAccount.setDraftsFolderName(getString(R.string.special_mailbox_name_drafts));
        mAccount.setTrashFolderName(getString(R.string.special_mailbox_name_trash));
        mAccount.setOutboxFolderName(getString(R.string.special_mailbox_name_outbox));
        mAccount.setSentFolderName(getString(R.string.special_mailbox_name_sent));
        AccountSetupAccountType.actionSelectAccountType(this, mAccount, true);
        finish();
    }
```

☑ 定义 onActivityResult()方法接收处理结果，当执行完 finish()后，Activity 执行结束，并且将返回值返回给调用它的父类 Activity 类。onActivityResult()方法第一个参数表示 Activity 请求码，第二个参数表示返回结果，结果码最常用的有 RESULT_OK 和 RESULT_CANCELED，前者表示执行成功，后者表示取消操作。对应部分代码如下所示。

```
   //根据指定返回码执行 Activity
    protected void onActivityResult(int requestCode, int resultCode,
                android.content.Intent data) {
```

```
            if (resultCode == RESULT_OK) {
                    mAccount.setDescription(mAccount.getEmail());
                    mAccount.save(Preferences.getPreferences(this));
                    Preferences.getPreferences(this).setDefaultAccount(mAccount);
                    AccountSetupNames.actionSetNames(this, mAccount);
                    finish();
               }
            }
public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            requestWindowFeature(Window.FEATURE_NO_TITLE);
            setContentView(R.layout.main);
            systemProvider=new SystemService(this);
            cursor=systemProvider.allSongs();
          //读 MUSIC 键值的值
            SharedPreferences sp = getSharedPreferences("MUSIC",MODE_WORLD_READABLE);
            if (sp != null) {
                  playingName = sp.getString("PLAYINGNAME", null);
                  selectName = sp.getString("SELECTNAME", null);
                  String s = sp.getString("MUSIC_LIST", null);
                  if (s != null)
                        music_List = StringHelper.spiltString(s);
            }
```

（2）系统主界面的布局文件是 activity_account_setup.xml，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">
<EditText
          android:id="@+id/account_email"
          android:hint="@string/account_setup_basics_email_hint"
          android:inputType="textEmailAddress"
          android:imeOptions="actionNext"
          android:layout_height="wrap_content"
          android:layout_width="fill_parent"
          />
<EditText
          android:id="@+id/account_password"
          android:hint="@string/account_setup_basics_password_hint"
          android:inputType="textPassword"
          android:imeOptions="actionDone"
          android:layout_height="wrap_content"
          android:layout_width="fill_parent"
          android:nextFocusDown="@+id/next"
/>
<View
          android:layout_width="fill_parent"
          android:layout_height="0px"
          android:layout_weight="1"
```

```
        />
    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="54dip"
        android:background="@android:drawable/menu_full_frame"
    >
    <Button
            android:id="@+id/next"
            android:text="@string/next_action"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:minWidth="100dip"
            android:drawableRight="@drawable/button_indicator_next"
            android:layout_alignParentRight="true"
            android:layout_centerVertical="true"
            />
    </RelativeLayout>
</LinearLayout>
```

☑ 以上代码中的 layout_height="match_parent"，其中 match_parent 和 fill_parent 的效果一样。

☑ nextFocusDown 定义单击 Down 键时，account_password 文本框获得焦点。nextFocusUp 定义单击 Up 键时，某组件获得焦点。nextFocusLeft 定义单击 Left 键时，某组件获得焦点。nextFocusRight 定义单击 Right 键时，某组件获得焦点。

☑ inputType 定义该组件是输入框类型。

☑ imeOptions 指定输入法窗口中的 Enter 键的功能，actionDone 表示软键盘下方变成“完成”，单击后光标保持在原来的输入框上，并且软键盘关闭。其他可选值为 normal、actionNext 和 actionSearch 等。

3.4.3 邮箱类型设置

在输入用户名和密码后，单击 next 按钮，将弹出邮箱类型设置窗口，如图 3-11 所示。

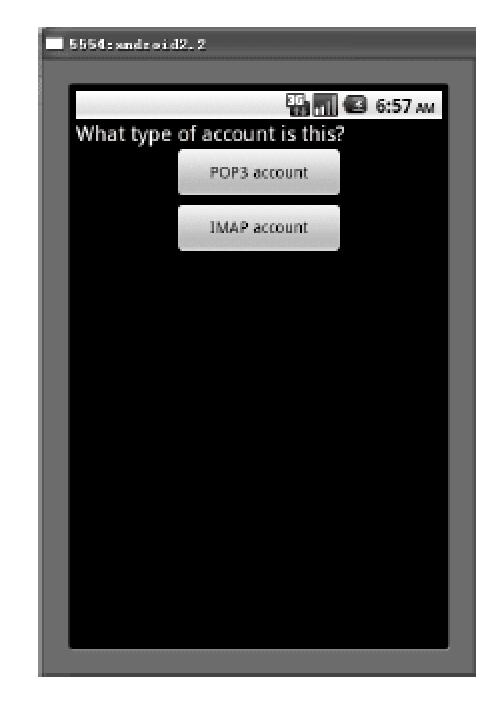
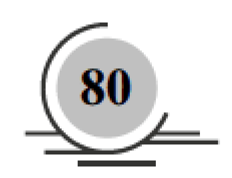


图 3-11　邮箱类型设置窗口

（1）编写文件 AccountSetupAccountType.java，在此定义邮箱类型设置界面。

☑ 定义 onCreate()方法初始化窗体，为 Button 对象定义监听器 setOnClickListener()。其中 Context 参数将接收从主界面窗体传送的数据，利用 actionSelectAccountType()方法做初始化操作，Intent()方法将程序执行跳转到 AccountSetupAccountType 实例。putExtra()方法以键值对的形式保存数据。对应部分代码如下所示。



```
public class AccountSetupAccountType extends Activity implements OnClickListener {
    private static final String EXTRA_ACCOUNT = "account";
    private static final String EXTRA_MAKE_DEFAULT = "makeDefault";
    private Account mAccount;
    private boolean mMakeDefault;
    //初始化
  public static void actionSelectAccountType(Context context, Account account, boolean makeDefault) {
        Intent i = new Intent(context, AccountSetupAccountType.class);
        //为 EXTRA_ACCOUNT 指定的键赋值
        i.putExtra(EXTRA_ACCOUNT, account);
        i.putExtra(EXTRA_MAKE_DEFAULT, makeDefault);
         //启动 Activity
        context.startActivity(i);
    }
   //创建一个窗体
   public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
         //加载 activity_account_setup_type 布局 XML 文件
        setContentView(R.layout.activity_account_setup_type);
        ((Button)findViewById(R.id.pop)).setOnClickListener(this);
        ((Button)findViewById(R.id.imap)).setOnClickListener(this);
        mAccount = (Account)getIntent().getSerializableExtra(EXTRA_ACCOUNT);
        mMakeDefault = (boolean)getIntent().getBooleanExtra(EXTRA_MAKE_DEFAULT, false);
    }
```

☑ 定义 onPop()方法保存用户的 URI 地址，getUserInfo()方法获得用户，getHost()方法获得主机地址，getPort()方法获得端口。此处的 URI 实例对象表示用户类型是 POP3 协议（允许用户从服务器上把邮件存储到本地主机）。对应部分代码如下所示。

```
    @Override
    private void onPop() {
        try {
        //定义并为 URI 实例赋值
            URI uri = new URI(mAccount.getStoreUri());
            uri = new URI("pop3", uri.getUserInfo(), uri.getHost(), uri.getPort(), null, null, null);
            mAccount.setStoreUri(uri.toString());
        } catch (URISyntaxException use) {
            throw new Error(use);
        }
        AccountSetupIncoming.actionIncomingSettings(this, mAccount, mMakeDefault);
        //执行 Activity
        finish();
    }
```

☑ 定义 onImap()方法保存用户的 URI 地址，此处的 URI 实例对象表示用户类型是 IMAP 协议（允许用户在线与邮件服务器交互信息）。无论是采用哪种通信协议，都要调用 actionIncomingSettings()方法进入用户收取邮件设置，onClick()监听用户单击的按钮动作。对应部分代码如下所示。

```
    private void onImap() {
        try {
        //定义并为 URI 实例赋值
        URI uri = new URI(mAccount.getStoreUri());
        uri = new URI("imap", uri.getUserInfo(), uri.getHost(), uri.getPort(), null, null, null);
```

```
                mAccount.setStoreUri(uri.toString());
            } catch (URISyntaxException use) {
                throw new Error(use);
            }
            mAccount.setDeletePolicy(Account.DELETE_POLICY_ON_DELETE);
            AccountSetupIncoming.actionIncomingSettings(this, mAccount, mMakeDefault);
            //执行 Activity
            finish();
        }
            //根据触发的组件调用相应的方法
        public void onClick(View v) {
            switch (v.getId()) {
                case R.id.pop:
                    onPop();
                    break;
                case R.id.imap:
                    onImap();
                    break;
            }
        }
```

（2）邮箱类型设置的布局文件是 activity_account_setup_type.xml，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
>
<TextView
        android:text="@string/account_setup_account_type_instructions"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="?android:attr/textColorPrimary"
        />
<Button
        android:id="@+id/pop"
        android:text="@string/account_setup_account_type_pop_action"
        android:layout_height="wrap_content"
        android:layout_width="150dip"
        android:minWidth="100dip"
        android:layout_gravity="center_horizontal"
        />
<Button
        android:id="@+id/imap"
        android:text="@string/account_setup_account_type_imap_action"
        android:layout_height="wrap_content"
        android:layout_width="150sp"
        android:minWidth="100dip"
        android:layout_gravity="center_horizontal"
        />
</LinearLayout>
```

☑ 以上代码中，android:textAppearance="?android:attr/textAppearanceMedium"引用的是系统自带的一个外观，“？”表示系统是否有这种外观，否则使用默认的外观。Android 的系统自带的文字外观设置及实际显示效果图可设置为 textappearancebutton、textappearanceinverse、textappearancelarge、textappearancelargeinverse、textappearancemedium、extappearancesmallinverse、textappearancemediuminverse 和 textappearancesmall。

☑ android:textColor="?android:attr/textColorPrimary"同样引用的是系统自带的一个外观。设置界面背景及文字颜色最常用的两种方法，即直接在布局文件中设置如 android:backgound="#FFFFFFFF", android:textcolor="#00000000"和把颜色提取出来形成资源，放在资源文件下面（如 values/drawable/color.xml）。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<drawable name="white">#FFFFFFFF</drawable>
<drawable name="black">#FF000000</drawable>
</resources>
```

然后在布局文件中使用 android:backgound="@drawable/white", android:textcolor="@drawable/black"或者在 Java 文件中通过 setBackgroundColor(int color), setBackgroundResource(int resid), setTextColor(int color)来设置背景颜色和文本颜色。

3.4.4　邮箱收取设置

在确定邮件类型后，单击 POP3 Account 或者 Imap Account 按钮，将弹出邮箱收取设置窗口，如图 3-12 所示。



图 3-12　邮箱收取设置窗口

此邮箱收取设置界面功能是通过文件 AccountSetupIncoming.java 实现的，接下来讲解此文件的实现流程。

☑ 定义 actionIncomingSettings()方法和 actionEditIncomingSettings()方法做数据初始化操作，其中，Context 参数将接收从主界面窗体传送的数据，Intent()方法将程序执行跳转到 AccountSetupIncoming 实例，putExtra()方法以键值对的形式保存数据。startActivity()方法启动在不同 Activity 间切换。对应部分代码如下所示。

```
private static final String EXTRA_ACCOUNT = "account";
    private static final String EXTRA_MAKE_DEFAULT = "makeDefault";
   //初始化端口选项
    private static final int popPorts[ ] = {
```

```
            110, 995, 995, 110, 110
    };
    private static final String popSchemes[ ] = {
            "pop3", "pop3+ssl", "pop3+ssl+", "pop3+tls", "pop3+tls+"
    };
    private static final int imapPorts[ ] = {
            143, 993, 993, 143, 143
    };
    private static final String imapSchemes[ ] = {
            "imap", "imap+ssl", "imap+ssl+", "imap+tls", "imap+tls+"
    };
    private int mAccountPorts[ ];
    private String mAccountSchemes[ ];
    private EditText mUsernameView;
    private EditText mPasswordView;
    private EditText mServerView;
    private EditText mPortView;
    private Spinner mSecurityTypeView;
    private Spinner mDeletePolicyView;
    private EditText mImapPathPrefixView;
    private Button mNextButton;
    private Account mAccount;
    private boolean mMakeDefault;
    public static void actionIncomingSettings(Activity context, Account account, boolean makeDefault) {
       //定义 Intent 对象，供 Activity 跳转
        Intent i = new Intent(context, AccountSetupIncoming.class);
        i.putExtra(EXTRA_ACCOUNT, account);
        i.putExtra(EXTRA_MAKE_DEFAULT, makeDefault);
        //启动 Activity(
        context.startActivity(i);
    }
  public static void actionEditIncomingSettings(Activity context, Account account) {
        Intent i = new Intent(context, AccountSetupIncoming.class);
        i.setAction(Intent.ACTION_EDIT);
        i.putExtra(EXTRA_ACCOUNT, account);
        context.startActivity(i);
  }
```

☑ Activity 之间进行切换。Android 开发中的四大组件包含活动（Activity）、服务（Services）、广播接收者（BroadcastReceiver）和内容提供者（ContentProvider）。其中，活动（Activity）是一个很重要的部分，表示一个可视化的用户界面，关注用户从事的事件，几乎所有的活动都要和用户进行交互。

☑ 定义 onCreate()方法初始化窗体，定义了 Spinner 控件，该控件主要就是一个列表。Spinner 是 View 类的一个子类，利用数组赋值的方式为其写入初值。对应部分代码如下所示。

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
       //加载 activity_account_setup_incoming 布局 XML 文件
        setContentView(R.layout.activity_account_setup_incoming);
        mUsernameView = (EditText)findViewById(R.id.account_username);
        mPasswordView = (EditText)findViewById(R.id.account_password);
        TextView serverLabelView = (TextView) findViewById(R.id.account_server_label);
```

```
        mServerView = (EditText)findViewById(R.id.account_server);
        mPortView = (EditText)findViewById(R.id.account_port);
        mSecurityTypeView = (Spinner)findViewById(R.id.account_security_type);
        mDeletePolicyView = (Spinner)findViewById(R.id.account_delete_policy);
        mImapPathPrefixView = (EditText)findViewById(R.id.imap_path_prefix);
        mNextButton = (Button)findViewById(R.id.next);
        //绑定监听器
        mNextButton.setOnClickListener(this);
        SpinnerOption securityTypes[ ] = {
                new SpinnerOption(0, getString(R.string.account_setup_incoming_security_none_label)),
                new SpinnerOption(1, getString(R.string.account_setup_incoming_security_ssl_optional_label)),
                new SpinnerOption(2, getString(R.string.account_setup_incoming_security_ssl_label)),
                new SpinnerOption(3, getString(R.string.account_setup_incoming_security_tls_optional_label)),
                new SpinnerOption(4, getString(R.string.account_setup_incoming_security_tls_label)),
        };
        SpinnerOption deletePolicies[ ] = {
                new SpinnerOption(0,getString(R.string.account_setup_incoming_delete_policy_never_label)),
                new SpinnerOption(1,getString(R.string.account_setup_incoming_delete_policy_7days_label)),
                new SpinnerOption(2,getString(R.string.account_setup_incoming_delete_policy_delete_label)),
        };
```

☑ ArrayAdapter 是从 BaseAdapter 派生出来的，具备 BaseAdapter 的所有功能，但 ArrayAdapter 更为强大，实例化时可以直接使用泛型构造。ArrayAdapter 分 3 种显示模式，分别是简单的、样式丰富但内容简单的和内容丰富的。Android SDK 中可以看到 android.widget.ArrayAdapter<T>形式，ArrayAdapter(Context context, int textViewResourceId) 第二个参数直接绑定一个 layout。对应部分代码如下所示。

```
ArrayAdapter<SpinnerOption> securityTypesAdapter = new ArrayAdapter<SpinnerOption>(this,
                android.R.layout.simple_spinner_item, securityTypes);
        securityTypesAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        mSecurityTypeView.setAdapter(securityTypesAdapter);  ArrayAdapter<SpinnerOption>  deletePolicies
Adapter = new ArrayAdapter<SpinnerOption>(this,
                android.R.layout.simple_spinner_item, deletePolicies);
        deletePoliciesAdapter
                .setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        mDeletePolicyView.setAdapter(deletePoliciesAdapter);
        mSecurityTypeView.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            public void onItemSelected(AdapterView arg0, View arg1, int arg2, long arg3) {
                updatePortFromSecurityType();
            }
            public void onNothingSelected(AdapterView<?> arg0) {
            }
});
```

☑ TextWatcher 实例监控 EditText 组件输入的内容发生的变化，然后定义 addTextChangedListener 监听器分别监控用户名、密码、服务和端口是否有输入内容，若没全部输入内容，则 next 按钮将成不可用状态。对应部分代码如下所示。

```
        TextWatcher validationTextWatcher = new TextWatcher() {
            public void afterTextChanged(Editable s) {
                validateFields();
            }
        };
```

```
    //定义监听器
     mUsernameView.addTextChangedListener(validationTextWatcher);
     mPasswordView.addTextChangedListener(validationTextWatcher);
     mServerView.addTextChangedListener(validationTextWatcher);
     mPortView.addTextChangedListener(validationTextWatcher);
     if (savedInstanceState != null && savedInstanceState.containsKey(EXTRA_ACCOUNT)) {
     //取得 mAccount 实例
         mAccount = (Account)savedInstanceState.getSerializable(EXTRA_ACCOUNT);
     }
     try {
         URI uri = new URI(mAccount.getStoreUri());
         String username = null;
         String password = null;
         if (uri.getUserInfo() != null) {
             String[ ] userInfoParts = uri.getUserInfo().split(":", 2);
             username = userInfoParts[0];
             if (userInfoParts.length > 1) {
                 password = userInfoParts[1];
             }
         }
         if (username != null) {
             mUsernameView.setText(username);
         }
         if (password != null) {
             mPasswordView.setText(password);
         }
```

☑ getScheme()方法返回当前请求所使用的协议，根据返回结果为 mAccountPorts 变量设置相应的值，对应部分代码如下所示。

```
 if
(uri.getScheme().startsWith("pop3"))
{serverLabelView.setText(R.string.account_setup_incoming_pop_server_label);
                 mAccountPorts = popPorts;
                 mAccountSchemes = popSchemes;findViewById(R.id.imap_path_prefix_section).setVisibility(View.GONE);
         } else if (uri.getScheme().startsWith("imap")) {serverLabelView.setText(R.string.account_setup_
incoming_imap_server_label);
                 mAccountPorts = imapPorts;
                 mAccountSchemes = imapSchemes;findViewById(R.id.account_delete_policy_label).setVisibility(View.GONE);
                 mDeletePolicyView.setVisibility(View.GONE);
                 if (uri.getPath() != null && uri.getPath().length() > 0) {
                     mImapPathPrefixView.setText(uri.getPath().substring(1));
                 }
         } else {
                 throw new Error("Unknown account type: " + mAccount.getStoreUri());
         }
         for (int i = 0; i < mAccountSchemes.length; i++) {
                 if (mAccountSchemes[i].equals(uri.getScheme())) {
                     SpinnerOption.setSpinnerOptionValue(mSecurityTypeView, i);
                 }
         }
         SpinnerOption.setSpinnerOptionValue(mDeletePolicyView, mAccount.getDeletePolicy());
         if (uri.getHost() != null) {
```

```
                mServerView.setText(uri.getHost());
            }
            if (uri.getPort() != -1) {
                mPortView.setText(Integer.toString(uri.getPort()));
            } else {
                updatePortFromSecurityType();
            }
        } catch (URISyntaxException use) {
            throw new Error(use);
        }
        //检查组件里内容的变化
        validateFields();
    }
```

（3）邮箱收取界面的布局文件是 activity_account_setup_incoming.xml，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:scrollbarStyle="outsideInset">
<LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical">

        <TextView
            android:id="@+id/account_server_label"
            android:text="@string/account_setup_incoming_pop_server_label"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:textColor="?android:attr/textColorPrimary" />
        <Spinner
            android:id="@+id/account_security_type"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent" />
        <TextView
            android:id="@+id/account_delete_policy_label"
            android:text="@string/account_setup_incoming_delete_policy_label"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:textColor="?android:attr/textColorPrimary" />
        <Spinner
            android:id="@+id/account_delete_policy"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent" />
        <View
            android:layout_width="fill_parent"
            android:layout_height="0px"
            android:layout_weight="1" />
        <RelativeLayout
```

```
                android:layout_width="fill_parent"
                android:layout_height="54dip"
                android:background="@android:drawable/menu_full_frame">
            <Button
                    android:id="@+id/next"
                    android:text="@string/next_action"
                    android:layout_height="wrap_content"
                    android:layout_width="wrap_content"
                    android:minWidth="100dip"
                    android:drawableRight="@drawable/button_indicator_next"
                    android:layout_alignParentRight="true"
                    android:layout_centerVertical="true" />
        </RelativeLayout>
    </LinearLayout>
</ScrollView>
```

☑ 以上代码中 android:drawableRight="@drawable/button_indicator_next"，Android SDK 中 Drawable 主要的作用是在 XML 中定义各种动画，然后把 XML 当作 Drawable 资源来读取，通过 Drawable 显示动画。

Drawable 就是一个可画的对象，可能是一张位图（BitmapDrawable），也可能是一个图形（ShapeDrawable），还有可能是一个图层（LayerDrawable），开发程序时为了兼容不同平台不同屏幕，所以要求建立多个文件夹，根据需求存放不同屏幕版本图片。

☑ 在 Android SDK 2.1 版本中有 drawable-mdpi、drawable-ldpi 和 drawable-hdpi 3 个文件夹，这 3 个文件夹主要是为了支持多分辨率。系统运行时会根据机器的分辨率分别到这几个文件夹中查找对应的图片。xhdpi 是从 Android 2.2（API Level 8）才开始增加的分类。xlarge 是从 Android 2.3（API Level 9）才开始增加的分类，在工程中 res/drawable-xhdpi/目录下存放 button_indicator_next.png 图片，另外，在以上 3 个目录中同样有此名字的图片。

3.4.5　邮箱发送设置

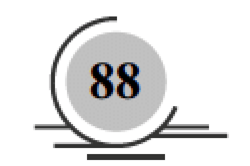
在设置好发送邮件的必要信息后，单击 Next 按钮，将弹出邮箱发送设置窗口，如图 3-13 所示。

（1）编写文件 AccountSetupOutgoing.java，在此定义邮箱发送设置界面，主要代码如下所示。

☑ 定义 actionOutgoingSettings()方法和 actionEditOutgoingSettings()方法，做数据初始化操作，其中，Context 参数将接收从主界面窗体传送的数据，Intent()方法将程序执行跳转到 AccountSetupOutgoing 实例，putExtra()方法以键值对的形式保存数据。startActivity()方法启动在不同 Activity 之间切换。主要代码如下所示。



图 3-13　邮箱发送设置窗口



```
public class AccountSetupOutgoing extends Activity implements OnClickListener,
          OnCheckedChangeListener {
     private static final String EXTRA_ACCOUNT = "account";
     private static final String EXTRA_MAKE_DEFAULT = "makeDefault";
    //定义发送端口
     private static final int smtpPorts[ ] = {
               25, 465, 465, 25, 25
     };
     private static final String smtpSchemes[ ] = {
               "smtp", "smtp+ssl", "smtp+ssl+", "smtp+tls", "smtp+tls+"
     };
     private EditText mUsernameView;
     private EditText mPasswordView;
     private EditText mServerView;
     private EditText mPortView;
     private CheckBox mRequireLoginView;
     private ViewGroup mRequireLoginSettingsView;
     private Spinner mSecurityTypeView;
     private Button mNextButton;
     private Account mAccount;
     private boolean mMakeDefault;
     public static void actionOutgoingSettings(Context context, Account account, boolean makeDefault) {
     //定义 Intent 实例，将 Activity 设定跳转到 AccountSetupOutgoing
          Intent i = new Intent (context, AccountSetupOutgoing.class);
          i.putExtra(EXTRA_ACCOUNT, account);
          i.putExtra(EXTRA_MAKE_DEFAULT, makeDefault);
          //启动 Activity
          context.startActivity(i);
     }
     public static void actionEditOutgoingSettings(Context context, Account account) {
          Intent i = new Intent(context, AccountSetupOutgoing.class);
          i.putExtra(EXTRA_ACCOUNT, account);
          context.startActivity(i);
     }
```

☑ 定义 onCreate(Bundle savedInstanceState)方法初始化窗体，创建 Activity 时调用。还以 Bundle 的形式提供对以前储存的任何状态的访问。主要代码如下所示。

```
     @Override
     public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          //加载界面布局 activity_account_setup_outgoing.xml 文件
          setContentView(R.layout.activity_account_setup_outgoing);
          mUsernameView = (EditText)findViewById(R.id.account_username);
          mPasswordView = (EditText)findViewById(R.id.account_password);
          mServerView = (EditText)findViewById(R.id.account_server);
          mPortView = (EditText)findViewById(R.id.account_port);
          mRequireLoginView = (CheckBox)findViewById(R.id.account_require_login);
          mRequireLoginSettingsView = (ViewGroup)findViewById(R.id.account_require_login_settings);
          mSecurityTypeView = (Spinner)findViewById(R.id.account_security_type);
          mNextButton = (Button)findViewById(R.id.next);
          mNextButton.setOnClickListener(this);
```

```
        //定义监听器
        mRequireLoginView.setOnCheckedChangeListener(this);
        SpinnerOption securityTypes[ ] = {
                new SpinnerOption(0, getString(R.string.account_setup_incoming_security_none_label)),
                new SpinnerOption(1,getString(R.string.account_setup_incoming_security_ssl_optional_label)),
                new SpinnerOption(2, getString(R.string.account_setup_incoming_security_ssl_label)),
                new SpinnerOption(3,getString(R.string.account_setup_incoming_security_tls_optional_label)),
                new SpinnerOption(4, getString(R.string.account_setup_incoming_security_tls_label)),
        };
        ArrayAdapter<SpinnerOption> securityTypesAdapter = new ArrayAdapter<SpinnerOption>(this,
                android.R.layout.simple_spinner_item, securityTypes);securityTypesAdapter.setDropDownViewResource
(android.R.layout.simple_spinner_dropdown_item);
        mSecurityTypeView.setAdapter(securityTypesAdapter);
        mSecurityTypeView.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            //树型组件单击后触发的事件
            public void onItemSelected(AdapterView arg0, View arg1, int arg2, long arg3) {
                updatePortFromSecurityType();
            }
            public void onNothingSelected(AdapterView<?> arg0) {
            }
        });
        TextWatcher validationTextWatcher = new TextWatcher() {
            public void afterTextChanged(Editable s) {
                validateFields();
            }
        };
        //定义监听器
        mUsernameView.addTextChangedListener(validationTextWatcher);
        mPasswordView.addTextChangedListener(validationTextWatcher);
        mServerView.addTextChangedListener(validationTextWatcher);
        mPortView.addTextChangedListener(validationTextWatcher);
        mPortView.setKeyListener(DigitsKeyListener.getInstance("0123456789"));
        mAccount = (Account)getIntent().getSerializableExtra(EXTRA_ACCOUNT);
        mMakeDefault = (boolean)getIntent().getBooleanExtra(EXTRA_MAKE_DEFAULT, false);
        if (savedInstanceState != null && savedInstanceState.containsKey(EXTRA_ACCOUNT)) {
            mAccount = (Account)savedInstanceState.getSerializable(EXTRA_ACCOUNT);
        }
        validateFields();
    }
private void validateFields() {
        boolean enabled =
            Utility.requiredFieldValid(mServerView) && Utility.requiredFieldValid(mPortView);
        if (enabled && mRequireLoginView.isChecked()) {
            enabled = (Utility.requiredFieldValid(mUsernameView)
                    && Utility.requiredFieldValid(mPasswordView));
        }
        if (enabled) {
            try {
                URI uri = getUri();
            } catch (URISyntaxException use) {
                enabled = false;
```

```
            }
        }
        mNextButton.setEnabled(enabled);
        Utility.setCompoundDrawablesAlpha(mNextButton, enabled ? 255 : 128);
    }
```

☑ 定义 TextWatcher 实例监控 EditText 组件输入的内容发生的变化。然后定义 addTextChangedListener 监听器分别监控用户名、密码、服务和端口是否有输入内容，若没全部输入内容，则 next 按钮将为不可用状态，主要代码如下所示。

```
 TextWatcher validationTextWatcher = new TextWatcher() {
            public void afterTextChanged(Editable s) {
                validateFields();
            }
 public void beforeTextChanged(CharSequence s, int start, int count, int after) {
            }
 public void onTextChanged(CharSequence s, int start, int before, int count) {
            }
        };
        mUsernameView.addTextChangedListener(validationTextWatcher);
        mPasswordView.addTextChangedListener(validationTextWatcher);
        mServerView.addTextChangedListener(validationTextWatcher);
        mPortView.addTextChangedListener(validationTextWatcher);
```

☑ 定义 updatePortFromSecurityType()方法，将用户输入的端口号赋值 mPortView 变量。代码 mSecurityTypeView.getSelectedItem().value 读取 View 的节点值，onActivityResult()方法检查 Activity 执行是否成功，主要代码如下所示。

```
private void updatePortFromSecurityType() {
        int securityType = (Integer)((SpinnerOption)mSecurityTypeView.getSelectedItem()).value;
        mPortView.setText(Integer.toString(smtpPorts[securityType]));
    }
    @Override
    //onActivityResult 执行完 Activity 后，将结果返回给调用它的父 Activity
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode == RESULT_OK) {
            if (Intent.ACTION_EDIT.equals(getIntent().getAction())) {
                mAccount.save(Preferences.getPreferences(this));
                finish();
            } else {
                AccountSetupOptions.actionOptions(this, mAccount, mMakeDefault);
                finish();
            }
        }
    }
    private URI getUri() throws URISyntaxException {
        int securityType = (Integer)((SpinnerOption)mSecurityTypeView.getSelectedItem()).value;
        String userInfo = null;
        if (mRequireLoginView.isChecked()) {
            userInfo = mUsernameView.getText().toString().trim() + ":"+
                      mPasswordView.getText().toString().trim();
        }
        URI uri = new URI(
```

```
                    smtpSchemes[securityType],
                    userInfo,
                    mServerView.getText().toString().trim(),
                    Integer.parseInt(mPortView.getText().toString().trim()),
                    null, null, null);
        return uri;
    }
```

☑ 定义 onClick()方法，用户单击 next 按钮后触发 onNext()方法。在该方法中读取邮件的 URI 地址，URI 地址包含有用户名和密码等信息。程序再跳转到 actionCheckSettings()方法中对用户的设置进行检查。

```
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.next:
                onNext();
                break;
        }
    }
private void onNext() {
        try {
            URI uri = getUri();
            mAccount.setSenderUri(uri.toString());
        } catch (URISyntaxException use) {
            throw new Error(use);
        }
        //检查用户
        AccountCheckSettings.actionCheckSettings(this, mAccount, false, true);
    }
```

（2）邮箱发送设置的布局文件是 AccountSetupOutgoing.xml，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:scrollbarStyle="outsideInset">
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical">
    <TextView
            android:text="@string/account_setup_outgoing_security_label"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:textColor="?android:attr/textColorPrimary" />
    <Spinner
            android:id="@+id/account_security_type"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent" />
    <CheckBox
            android:id="@+id/account_require_login"
```

```
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/account_setup_outgoing_require_login_label" />
    <LinearLayout
            android:id="@+id/account_require_login_settings"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:orientation="vertical"
            android:visibility="gone">
    </LinearLayout>
</ScrollView>
```

☑ 以上代码中，android:scrollbarStyle="outsideInset"引用的是系统自带的一个外观，“？”表示系统是否有这种外观，否则使用默认的外观。Android 系统自带的文字外观设置及实际显示效果图可设置为 textappearancebutton、textappearanceinverse、textappearancelarge、textappearancelargeinverse、textappearancemedium、extappearancesmallinverse、textappearancemediuminverse 和 textappearancesmall。

3.4.6　邮箱用户检查

在设置好邮件后，单击 next 按钮，将弹出邮箱用户检查窗口，如图 3-14 所示。

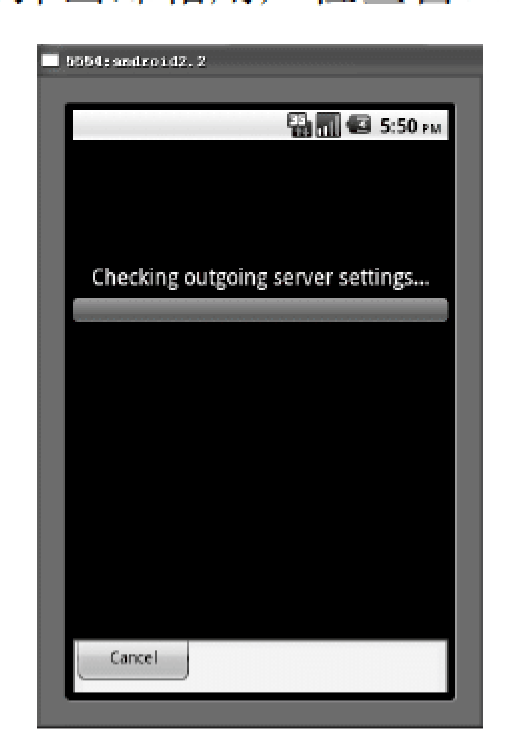


图 3-14　邮箱用户检查窗口

（1）编写文件 AccountCheckSettings.java，在此定义邮箱用户检查界面。

☑ 定义 actionCheckSettings()方法做数据初始化操作，startActivityForResult()方法的第一个参数是一个 Intent；第二个参数是返回码。通过方法不同的返回码，可以区分不同的 Activity，当启动了某个 Activity 后，返回码依然关联着当前进程所处理的 Activity。当操作完成后，会有特定的返回值响应某些事件。即 Activity 执行 finish()以后执行回调方法 onActivityResult()，而使用 startActivity()方法却不会执行回调。对应代码如下所示。

```
public class AccountCheckSettings extends Activity implements OnClickListener {
    private static final String EXTRA_ACCOUNT = "account";
    private static final String EXTRA_CHECK_INCOMING = "checkIncoming";
    private static final String EXTRA_CHECK_OUTGOING = "checkOutgoing";
    private Handler mHandler = new Handler();
    private ProgressBar mProgressBar;
    private TextView mMessageView;
    private Account mAccount;
    private boolean mCheckIncoming;
```

```
private boolean mCheckOutgoing;
private boolean mCanceled;
private boolean mDestroyed;
public static void actionCheckSettings(Activity context, Account account,
        boolean checkIncoming, boolean checkOutgoing) {
    Intent i = new Intent(context, AccountCheckSettings.class);
   //为 Account 对象的键名赋值
    i.putExtra(EXTRA_ACCOUNT, account);
    i.putExtra(EXTRA_CHECK_INCOMING, checkIncoming);
    i.putExtra(EXTRA_CHECK_OUTGOING, checkOutgoing);
   //指定将要启动的 Activity 的编号
    context.startActivityForResult(i, 1);
}
```

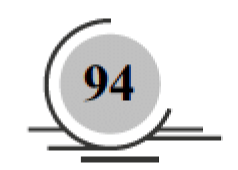
☑ 定义 onCreate()方法初始化窗体，定义了 mProgressBar 控件，该控件是一个进度条。mProgressBar 对象的 setIndeterminate()方法开启滚动效果。getIntent()方法获得当前的 Intent 的信息实例，然后调用各 get 方法获取对应的值。对应部分代码如下所示。

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_account_check_settings);
    mMessageView = (TextView)findViewById(R.id.message);
    mProgressBar = (ProgressBar)findViewById(R.id.progress);
    ((Button)findViewById(R.id.cancel)).setOnClickListener(this);
    setMessage(R.string.account_setup_check_settings_retr_info_msg);
    //打开进度条的滚动效果
    mProgressBar.setIndeterminate(true);
    mAccount = (Account)getIntent().getSerializableExtra(EXTRA_ACCOUNT);
    mCheckIncoming = (boolean)getIntent().getBooleanExtra(EXTRA_CHECK_INCOMING, false);
    mCheckOutgoing = (boolean)getIntent().getBooleanExtra(EXTRA_CHECK_OUTGOING, false);
```

☑ 实例化一个线程 Thread，setThreadPriority()方法设置线程在后台执行。程序开始要针对用户的发送邮件进行检查，如果当前 Activity 状态处于 Destroyed 和 Canceled，线程退出。对象 Sender 调用方法 getInstance()读取用户的 URI 信息，然后调用 open()方法打开地址。如果能够顺利地完成一次 close()和 open()方法的操作，并且不报异常，说明用户提供的地址可以正确使用，如图 3-15 所示。部分代码如下所示。



图 3-15　用户检查成功



```
        new Thread() {
            public void run() {
                //设置线程执行级别
                Process.setThreadPriority(Process.THREAD_PRIORITY_BACKGROUND);
                try {
                    if (mDestroyed) {
                        return;
                    }
                    if (mCanceled) {
                        finish();
                        return;
                    }
                    if (mCheckIncoming) {
                        setMessage(R.string.account_setup_check_settings_check_incoming_msg);
                    }
                    if (mDestroyed) {
                        return;
                    }
                    if (mCanceled) {
                        finish();
                        return;
                    }
                    if (mCheckOutgoing) {
                        setMessage(R.string.account_setup_check_settings_check_outgoing_msg);
                        Sender sender = Sender.getInstance(mAccount.getSenderUri());
                        sender.close();
                        sender.open();
                        sender.close();
                    }
                    if (mDestroyed) {
                        return;
                    }
                    if (mCanceled) {
                        finish();
                        return;
                    }
                    setResult(RESULT_OK);
                    finish();
```

☑ 在此定义捕获AuthenticationFailedException类型、CertificateValidationException类型和Messaging Exception类型的异常。对应部分代码如下所示。

```
                 catch (final AuthenticationFailedException afe) {
                    String message = afe.getMessage();
                    int id = (message == null)
                            ? R.string.account_setup_failed_dlg_auth_message
                            : R.string.account_setup_failed_dlg_auth_message_fmt;
                     //显示错误信息
                    showErrorDialog(id, message);
                } catch (final CertificateValidationException cve) {
                    String message = cve.getMessage();
                    int id = (message == null)
```

```
                        ? R.string.account_setup_failed_dlg_certificate_message
                        : R.string.account_setup_failed_dlg_certificate_message_fmt;
                    showErrorDialog(id, message);
                } catch (final MessagingException me) {
                    int id;
                    String message = me.getMessage();
                    switch (me.getExceptionType()) {
                        case MessagingException.IOERROR:
                            id = R.string.account_setup_failed_ioerror;
                            break;
                        case MessagingException.TLS_REQUIRED:
                            id = R.string.account_setup_failed_tls_required;
                            break;
                        case MessagingException.AUTH_REQUIRED:
                            id = R.string.account_setup_failed_auth_required;
                            break;
                        case MessagingException.GENERAL_SECURITY:
                            id = R.string.account_setup_failed_security;
                            break;
                        default:
                            id = (message == null)
                                    ? R.string.account_setup_failed_dlg_server_message
                                    : R.string.account_setup_failed_dlg_server_message_fmt;
                            break;
                    }
                    showErrorDialog(id, message);
                }
            }
        }.start();
    }
```

☑ 定义 showErrorDialog()方法显示错误对话框，显示具体内容在 AlertDialog 实例中指定。同时中止进度条的滚动显示 setIndeterminate(false)，对应部分代码如下所示。

```
private void showErrorDialog(final int msgResId, final Object... args) {
        mHandler.post(new Runnable() {
            public void run() {
                if (mDestroyed) {
                    return;
                }
                //关闭进度条的滚动效果
                mProgressBar.setIndeterminate(false);
                //创建一个提示对话框
                new AlertDialog.Builder(AccountCheckSettings.this)
                        .setIcon(android.R.drawable.ic_dialog_alert)
                        .setTitle(getString(R.string.account_setup_failed_dlg_title))
                        .setMessage(getString(msgResId, args))
                        .setCancelable(true)
                        .setPositiveButton(getString(R.string.account_setup_failed_dlg_edit_details_action),
                                new DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface dialog, int which) {
                                        finish();
```

```
                                                }
                                        })
                                .show();
                }
            });
    }
  private void onCancel() {
            mCanceled = true;
            setMessage(R.string.account_setup_check_settings_canceling_msg);
    }
    public void onClick(View v) {
            switch (v.getId()) {
                    case R.id.cancel:
                            onCancel();
                            break;
            }
    }
```

（2）设置用户别名的布局文件是 activity_account_setup_incoming.xml，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <View
        android:layout_width="fill_parent"
        android:layout_height="100sp" />
    <TextView
        android:id="@+id/message"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:gravity="center_horizontal"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="?android:attr/textColorPrimary"
        android:paddingBottom="6px" />
    <ProgressBar
        android:id="@+id/progress"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        style="?android:attr/progressBarStyleHorizontal" />
    <View
        android:layout_width="fill_parent"
        android:layout_height="0px"
        android:layout_weight="1" />
    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="54dip"
        android:background="@android:drawable/menu_full_frame">
    <Button
            android:id="@+id/cancel"
            android:text="@string/cancel_action"
```

```
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:minWidth="100dip"
            android:layout_centerVertical="true" />
    </RelativeLayout>
</LinearLayout>
```

以上代码中有 android:backgrund="@android:drawable/menu_full_frame"，Android SDK 中 Drawable 主要的作用是在 XML 中定义各种动画，然后把 XML 当作 Drawable 资源来读取，通过 Drawable 显示动画。

3.4.7 设置用户别名

在成功通过地址检测后，单击 next 按钮，将弹出设置用户别名窗口，如图 3-16 所示。

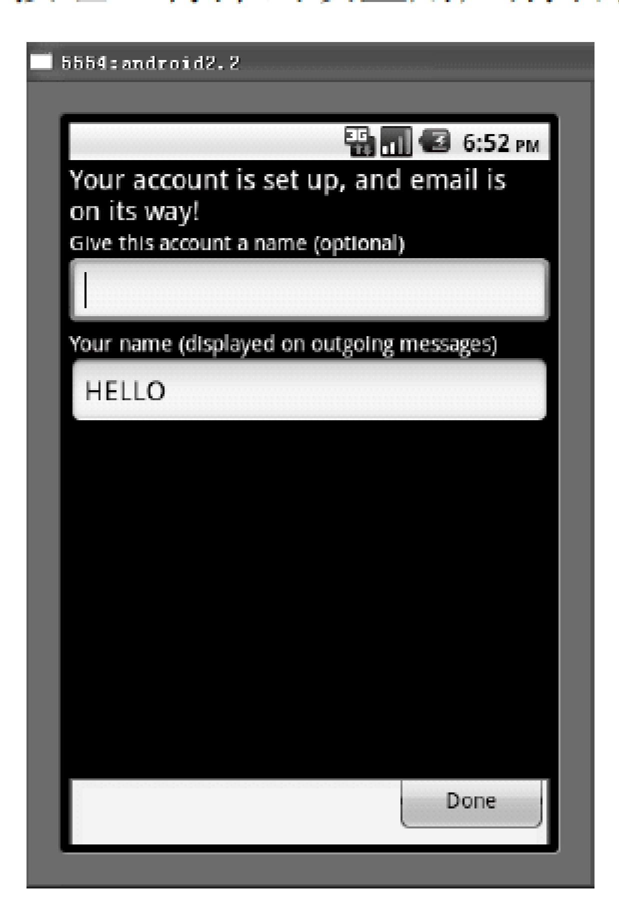


图 3-16 设置用户别名窗口

（1）编写文件 AccountSetupNames.java，在此定义设置用户别名界面。

☑ 定义 actionSetNames()方法做数据初始化操作，startActivity()方法启动 Activity。onCreate()方法定义两个文本框组件，还定义 setOnClickListener 监听器，validateFields()方法一旦发现文本框中的内容都发生变化后，执行监听器中的动作。对应部分代码如下所示。

```
public class AccountSetupNames extends Activity implements OnClickListener {
    private static final String EXTRA_ACCOUNT = "account";
    private EditText mDescription;
    private EditText mName;
    private Account mAccount;
    private Button mDoneButton;
    public static void actionSetNames(Context context, Account account) {
        //为 AccountSetupNames 窗体的 Activity 赋值
        Intent i = new Intent(context, AccountSetupNames.class);
        i.putExtra(EXTRA_ACCOUNT, account);
      //启动 Account 窗体
       context.startActivity(i);
    }
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_account_setup_names);
        mDescription = (EditText)findViewById(R.id.account_description);
        mName = (EditText)findViewById(R.id.account_name);
        mDoneButton = (Button)findViewById(R.id.done);
        //定义监听器
        mDoneButton.setOnClickListener(this);
        TextWatcher validationTextWatcher = new TextWatcher() {
            public void afterTextChanged(Editable s) {
                validateFields();
            }
        };
```

☑ 调用 mName.setText()方法保存用户名称。onNext()方法将当前设置的用户名保存在 Preferences 对象中，然后 Activity 跳转到邮件编辑界面 EmailCpsActivity。对应部分代码如下所示。

```
        mName.addTextChangedListener(validationTextWatcher);
        mName.setKeyListener(TextKeyListener.getInstance(false, Capitalize.WORDS));
        mAccount = (Account)getIntent().getSerializableExtra(EXTRA_ACCOUNT);
        // mDescription.setText(mAccount.getDescription());
        if (mAccount.getName() != null) {
            mName.setText(mAccount.getName());
        }
        if (!Utility.requiredFieldValid(mName)) {
            mDoneButton.setEnabled(false);
        }
    }
    private void validateFields() {
        mDoneButton.setEnabled(Utility.requiredFieldValid(mName));
        Utility.setCompoundDrawablesAlpha(mDoneButton, mDoneButton.isEnabled() ? 255 : 128);
    }
    private void onNext() {
        if (Utility.requiredFieldValid(mDescription)) {
            mAccount.setDescription(mDescription.getText().toString());
        }
        mAccount.setName(mName.getText().toString());
        mAccount.save(Preferences.getPreferences(this));
        //定义一个 Activity 名为 EmailCpsActivity
          Intent intent = new Intent(this, EmailCpsActivity.class);
        intent.putExtra(EXTRA_ACCOUNT , mAccount);
        //启动 Activity
        startActivity(intent);
        //执行
        finish();
    }
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.done:
                onNext();
                break;
        }
    }
}
```

（2）设置用户别名的布局文件是 activity_account_setup_names.xml ，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <TextView
        android:text="@string/account_setup_names_instructions"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="?android:attr/textColorPrimary"/>
    <TextView
        android:text="@string/account_setup_names_account_name_label"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:textColor="?android:attr/textColorPrimary"/>
    <EditText
        android:id="@+id/account_description"
        android:inputType="textCapWords"
        android:imeOptions="actionDone"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"/>
    <TextView
        android:text="@string/account_setup_names_user_name_label"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:textColor="?android:attr/textColorPrimary"/>
    <EditText
        android:id="@+id/account_name"
        android:inputType="textPersonName"
        android:imeOptions="actionDone"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"/>
    <View
        android:layout_height="0px"
        android:layout_width="fill_parent"
        android:layout_weight="1"/>
    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="54dip"
        android:background="@android:drawable/menu_full_frame">
    <Button
        android:id="@+id/done"
        android:text="@string/done_action"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:minWidth="100dip"
```

```
                android:layout_alignParentRight="true"
                android:layout_centerVertical="true"/>
    </RelativeLayout>
</LinearLayout>
```

☑ 以上代码中的 android:inputType="textCapWords"，inputType 用于设置键盘类型，其参数有 textcapcharacters（字母大写）、numbersigned（有符号数字格式）、textcapwords（单词首字母大写）、textcapsentences（仅第一个字母大写）、textautocomplete（自动完成）、textmultiline（多行输入）、textimemultiline（输入法多行）、textnosuggestions（不提示）、textemailaddress（电子邮件地址）、textemailsubject（邮件主题）、textshortmessage（短信息）、textpersonname（人名）、textpostaladdress（地址）、textpassword （密码）、textvisiblepassword（可见密码）、textwebedittext（作为网页表单的文本）、textfilte（文本筛选过滤）和 textphonetic（拼音输入）。

3.4.8　用户邮件编辑

在设置完别名后，单击 next 按钮，将弹出用户邮件编辑窗口，如图 3-17 所示。

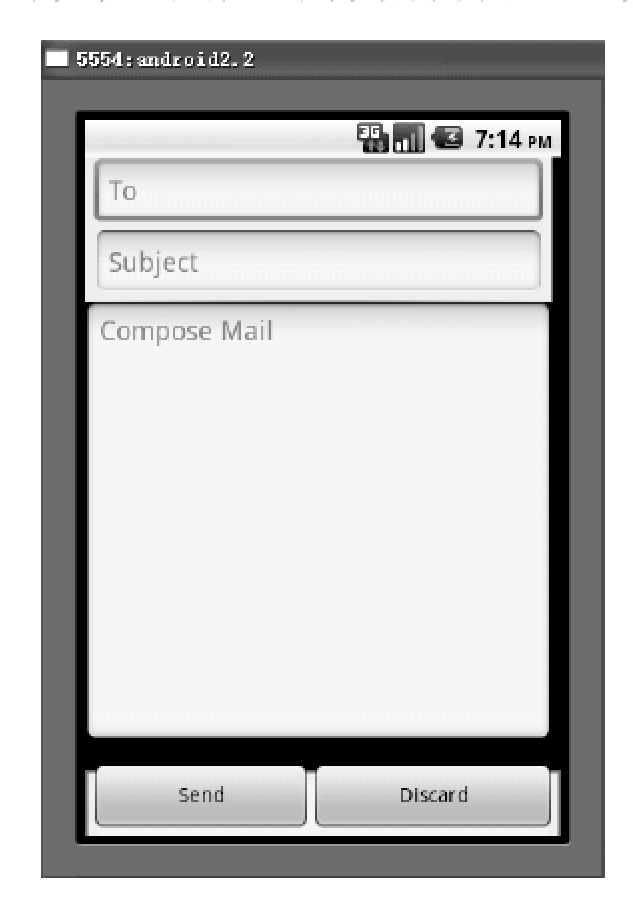


图 3-17　用户邮件编辑窗口

（1）编写文件 EmailCpsActivity.java，在此定义用户邮件编辑界面，主要代码如下所示。

☑ 定义 Handler 实例化对象，每一个 Handler 类都和一个唯一的线程（以及这个线程的 MessageQueue）关联。向其所关联的 MessageQueue 递送 Messages/Runnables，主要用途是按计划发送消息或执行某个 Runnanble（使用 POST 方法）；从其他线程中发送来的消息放入消息队列中，避免线程冲突（常见于更新 UI 线程）。对应部分代码如下所示。

```
public class EmailCpsActivity extends Activity implements OnClickListener, OnFocusChangeListener {
    private static final String EXTRA_ACCOUNT = "account";
    private static final int MSG_PROGRESS_ON = 1;
    private static final int MSG_PROGRESS_OFF = 2;
    private static final int MSG_UPDATE_TITLE = 3;
    private static final int MSG_SKIPPED_ATTACHMENTS = 4;
    private static final int MSG_SAVED_DRAFT = 5;
    private static final int MSG_DISCARDED_DRAFT = 6;
    private Account mAccount;
    private MultiAutoCompleteTextView mToView;
    private MultiAutoCompleteTextView mCcView;
```

```
    private MultiAutoCompleteTextView mBccView;
    private EditText mSubjectView;
    private EditText mMessageContentView;
    private Button mSendButton;
    private Button mDiscardButton;
    private ProgressDialog progress;
    private Handler mHandler = new Handler() {
        @Override
        public void handleMessage(android.os.Message msg) {
            switch (msg.what) {
                case MSG_PROGRESS_ON:
                    //开启滚动条的滚动效果
                    setProgressBarIndeterminateVisibility(true);
                    break;
                case MSG_PROGRESS_OFF:
                    //关闭滚动条的滚动效果
                    setProgressBarIndeterminateVisibility(false);
                    break;
                case MSG_UPDATE_TITLE:
                    updateTitle();
                    break;
                case MSG_SKIPPED_ATTACHMENTS:
                    Toast.makeText(
                            EmailCpsActivity.this,
getString(R.string.message_compose_attachments_skipped_toast),
                            Toast.LENGTH_LONG).show();
                    break;
                case MSG_SAVED_DRAFT:
                    Toast.makeText(
                            EmailCpsActivity.this,
                            getString(R.string.message_saved_toast),
                            Toast.LENGTH_LONG).show();
                    break;
                case MSG_DISCARDED_DRAFT:
                    Toast.makeText(
                            EmailCpsActivity.this,
                            getString(R.string.message_discarded_toast),
                            Toast.LENGTH_LONG).show();
                    break;
                default:
                    super.handleMessage(msg);
                    break;
            }
        }
    };
    private Validator mAddressValidator;
    @Override
```

☑ 定义 onCreate()方法做数据初始化操作，定义一个 MultiAutoCompleteTextView 对象，该对象继承自 AutoCompleteTextView 的可编辑的文本视图，能够对用户输入的文本进行有效的扩充提示，而不需要用户输入整个内容。

☑ requestWindowFeature()方法的功能是启用窗体的扩展特性。参数是 Window 类中定义的常量。
- DEFAULT_FEATURES：系统默认状态，一般不需要指定。
- FEATURE_CONTEXT_MENU：启用 ContextMenu，默认该项已启用，一般无需指定。
- FEATURE_CUSTOM_TITLE：自定义标题。当需要自定义标题时必须指定。例如，标题是一个按钮时。
- FEATURE_INDETERMINATE_PROGRESS：不确定的进度。
- FEATURE_LEFT_ICON：标题栏左侧的图标。
- FEATURE_NO_TITLE：无标题。
- FEATURE_OPTIONS_PANEL：启用“选项面板”功能，默认已启用。
- FEATURE_PROGRESS：进度指示器功能。
- FEATURE_RIGHT_ICON：标题栏右侧的图标。

对应部分代码如下所示。

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
    //加载 activity_compose.xml 布局界面文件
    setContentView(R.layout.activity_compose);
    mAddressValidator = new EmailAddressValidator();
    mToView = (MultiAutoCompleteTextView)findViewById(R.id.to);
    mCcView = (MultiAutoCompleteTextView)findViewById(R.id.cc);
    mBccView = (MultiAutoCompleteTextView)findViewById(R.id.bcc);
    mSubjectView = (EditText)findViewById(R.id.subject);
    mMessageContentView = (EditText)findViewById(R.id.message_content);
    mSendButton = (Button)findViewById(R.id.send);
    mDiscardButton = (Button)findViewById(R.id.discard);
```

☑ 定义 InputFilter 对象的实例 recipientFilter 使用输入过滤器 InputFilter 约束用户输入。定义一个返回类型为 CharSequence 的方法 filter()，检查用户的所有输入是否合法。对应部分代码如下所示。

```
    InputFilter recipientFilter = new InputFilter() {
//定义字符过滤方法
public CharSequence filter(CharSequence source, int start, int end, Spanned dest,
            int dstart, int dend) {
        if (end-start != 1 || source.charAt(start) != ' ') {
            return null;
        }
        int scanBack = dstart;
        boolean dotFound = false;
        while (scanBack > 0) {
            char c = dest.charAt(--scanBack);
            switch (c) {
                case '.':
                    dotFound = true;    //one or more dots are req'd
                    break;
                case ',':
                    return null;
                case '@':
                    if (!dotFound) {
                        return null;
                    }
```

```
                    if (source instanceof Spanned) {
                        SpannableStringBuilder sb = new SpannableStringBuilder(",");
                        sb.append(source);
                        return sb;
                    } else {
                        return ", ";
                    }
                default:
                    //just keep going
            }
        }
        //no termination cases were found, so don't edit the input
        return null;
    }
};
InputFilter[ ] recipientFilters = new InputFilter[ ] { recipientFilter };
//NOTE: assumes no other filters are set
```

☑ 将输入过滤器作用于 mToView、mCcView、mBccView、mToView、mToView 和 mCcView 组件之上。setOnFocusChangeListener()方法将焦点定位于该组件上，为发送和取消按钮定义 setOnClickListener 监听器。对应部分代码如下所示。

```
    //为组件指定过滤规则
    mToView.setFilters(recipientFilters);
    mCcView.setFilters(recipientFilters);
    mBccView.setFilters(recipientFilters);
    mToView.setTokenizer(new Rfc822Tokenizer());
    mToView.setValidator(mAddressValidator);
    mCcView.setTokenizer(new Rfc822Tokenizer());
    mCcView.setValidator(mAddressValidator);
    mBccView.setTokenizer(new Rfc822Tokenizer());
    mBccView.setValidator(mAddressValidator);
    mSendButton.setOnClickListener(this);
    mDiscardButton.setOnClickListener(this);
    mSubjectView.setOnFocusChangeListener(this);
    Intent intent = getIntent();
    mAccount = (Account) intent.getSerializableExtra(EXTRA_ACCOUNT);
    updateTitle();
}
@Override
//暂停后恢复调用
public void onResume() {
    super.onResume();
}
@Override
//暂停
public void onPause() {
    super.onPause();
}
@Override
//退出
public void onDestroy() {
```

```
        super.onDestroy();
    }
    private void updateTitle() {
        if (mSubjectView.getText().length() == 0) {
            setTitle(R.string.compose_title);
        } else {
            setTitle(mSubjectView.getText().toString());
        }
    }
    //焦点发生变化
    public void onFocusChange(View view, boolean focused) {
        if (!focused) {
            updateTitle();
        }
    }
    private Address[ ] getAddresses(MultiAutoCompleteTextView view) {
        Address[ ] addresses = Address.parse(view.getText().toString().trim());
        return addresses;
    }
```

☑ 定义一个 MimeMessage 对象 message 实例，类 MimeMessage 继承自定义类 Message，封装邮件发送时的信息。调用 setFrom()方法、setRecipients()方法和 setSubject()方法进行邮件信息头封装操作。再定义一个 TextBody 对象 body 实例，调用 setBody()方法将文本内容写入其中。对应部分代码如下所示。

```
    private MimeMessage createMessage() throws MessagingException {
        MimeMessage message = new MimeMessage();
        message.setSentDate(new Date());
        Address from = new Address(mAccount.getEmail(), mAccount.getName());
        message.setFrom(from);
        message.setRecipients(RecipientType.TO, getAddresses(mToView));
        message.setRecipients(RecipientType.CC, getAddresses(mCcView));
        message.setRecipients(RecipientType.BCC, getAddresses(mBccView));
        message.setSubject(mSubjectView.getText().toString());
        String text = mMessageContentView.getText().toString();
        //打印输出日志信息
        Log.d(Email.LOG_TAG, text);
        TextBody body = new TextBody(text);
        message.setBody(body);
        return message;
    }
    private void sendMessage() {
        //定义一个进度条
        progress = ProgressDialog.show(this, "", "sending...");
        final MimeMessage message;
        try {
            message = createMessage();
        }
        catch (MessagingException me) {
           //打印输出日志信息
            Log.e(Email.LOG_TAG, "Failed to create new message for send or save.", me);
            throw new RuntimeException("Failed to create a new message for send or save.", me);
```

```
        }
        Thread thread = new Thread(new Runnable(){
            @Override
            public void run() {
            try {
            Sender sender = Sender.getInstance(mAccount.getSenderUri());
            ArrayList<Part> viewables = new ArrayList<Part>();
                    ArrayList<Part> attachments = new ArrayList<Part>();
                    MimeUtility.collectParts(message, viewables, attachments);
                    StringBuffer sbHtml = new StringBuffer();
                    StringBuffer sbText = new StringBuffer();
                    for (Part viewable : viewables) {
                        try {
                            String text = MimeUtility.getTextFromPart(viewable);
                            /*
                            * Anything with MIME type text/html will be stored as such. Anything
                             * else will be stored as text/plain.
                             */
                            if (viewable.getMimeType().equalsIgnoreCase("text/html")) {
                                sbHtml.append(text);
                            }
                            else {
                                sbText.append(text);
                            }
                        } catch (Exception e) {
      throw new MessagingException("Unable to get text for message part", e);
                        }
                    }
                    message.setUid("email" + UUID.randomUUID().toString());
       message.setHeader(MimeHeader.HEADER_CONTENT_TYPE, "multipart/mixed");
                    MimeMultipart mp = new MimeMultipart();
                    mp.setSubType("mixed");
                    message.setBody(mp);
                      String htmlContent = sbHtml.toString();
                      String textContent = sbText.toString();
                      if (htmlContent != null) {
                           TextBody body = new TextBody(htmlContent);
                           MimeBodyPart bp = new MimeBodyPart(body, "text/html");
                           mp.addBodyPart(bp);
                           Log.v(Email.LOG_TAG, htmlContent);
                      }
                      if (textContent != null) {
                           TextBody body = new TextBody(textContent);
                           MimeBodyPart bp = new MimeBodyPart(body, "text/plain");
                           mp.addBodyPart(bp);
                           Log.v(Email.LOG_TAG, textContent);
                      }
              //发送信息
              sender.sendMessage(message);
                   } catch (MessagingException e) {
                      e.printStackTrace();
```

```
                }
                progress.dismiss();
                finish();
            }
        });
        thread.start();
    }
```

☑ 对象 Toast 是 Android 中用来显示信息的一种机制，与 Dialog 有所不同，Toast 没有焦点，而且显示的时间很短，在一定的时间就会自动消失。sendMessage()正式进行邮件发送操作。对应部分代码如下所示。

```
    private void onSend() {
        if (getAddresses(mToView).length == 0 &&
                getAddresses(mCcView).length == 0 &&
                getAddresses(mBccView).length == 0) {
            mToView.setError(getString(R.string.message_compose_error_no_recipients));
            Toast.makeText(this, getString(R.string.message_compose_error_no_recipients),
                    Toast.LENGTH_LONG).show();
            return;
        }
        sendMessage();
    }
    //发送信息主方法
    private void onDiscard() {
        mHandler.sendEmptyMessage(MSG_DISCARDED_DRAFT);
        finish();
    }
    public void onClick(View view) {
        switch (view.getId()) {
            case R.id.send:
                onSend();
                break;
            case R.id.discard:
                onDiscard();
                break;
        }
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.send:
                onSend();
                break;
            case R.id.discard:
                onDiscard();
                break;
            default:
                return super.onOptionsItemSelected(item);
        }
        return true;
    }
}
```

（2）用户邮件编辑的布局文件是 activity_compose.xml，主要代码如下所示。

☑ 下面代码中 android:scrollbarStyle="outsideInset"用于设置滚动条的风格。

☑ android:fillViewport="true"，当定义 scrollview 的子控件不足 scrollbarStyle 大小时，对其设定的 fill_parent 属性时不起作用，此时必须加 fillviewport 属性。对应部分代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent" android:layout_width="fill_parent"
    android:orientation="vertical">
    <ScrollView android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:layout_weight="1"
        android:scrollbarStyle="outsideInset"
        android:fillViewport="true">
        <LinearLayout android:orientation="vertical"
            android:layout_width="fill_parent" android:layout_height="wrap_content">
            <LinearLayout android:orientation="vertical"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content" android:background="#ededed">
                <MultiAutoCompleteTextView
                    android:id="@+id/to" android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:textAppearance="?android:attr/textAppearanceMedium"
                    android:textColor="?android:attr/textColorSecondaryInverse"
                    android:layout_marginLeft="6px"
                    android:layout_marginRight="6px"
                    android:inputType="textEmailAddress|textMultiLine"
                    android:imeOptions="actionNext"
                    android:hint="@string/message_compose_to_hint" />
```

☑ android:hint="@string/message_compose_bcc_hint"，设置 EditText 为空时的提示信息。

☑ android:visibility="gone"，表示此视图是否显示。3 个属性分别是 visible（显示）、invisible（显示黑背景条）和 gone（不显示）。对应部分代码如下所示。

```
<MultiAutoCompleteTextView
                    android:id="@+id/cc" android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:textAppearance="?android:attr/textAppearanceMedium"
                    android:textColor="?android:attr/textColorSecondaryInverse"
                    android:layout_marginLeft="6px"
                    android:layout_marginRight="6px"
                    android:inputType="textEmailAddress|textMultiLine"
                    android:imeOptions="actionNext"
                    android:hint="@string/message_compose_cc_hint"
                    android:visibility="gone" />
              <MultiAutoCompleteTextView
                    android:id="@+id/bcc" android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
android:textAppearance="?android:attr/textAppearanceMedium"
                    android:textColor="?android:attr/textColorSecondaryInverse"
                    android:layout_marginLeft="6px"
                    android:layout_marginRight="6px"
```

```
                    android:inputType="textEmailAddress|textMultiLine"
                    android:imeOptions="actionNext"
                    android:hint="@string/message_compose_bcc_hint"
                    android:visibility="gone" />
```

☑ android:inputType="textEmailAddress｜textMultiLine"，用于设置键盘输入类型。当多种类型同时定义时用"｜"分隔符，例如 android:inputType="textEmailSubject｜textAutoCorrect｜textCapSentences｜textImeMultiLine"。

☑ android:gravity="top"，设置控件上信息的位置，参数有 center（居中）、bottom（下）、top（上）、right（右）和 left（左），如定义左下的效果 android:gravity=" left| bottom "。对应部分代码如下所示。

```
<EditText android:id="@+id/subject"
                    android:layout_width="fill_parent"
                    android:textAppearance="?android:attr/textAppearanceMedium"
                    android:layout_height="wrap_content"
                    android:textColor="?android:attr/textColorSecondaryInverse"
                    android:layout_marginLeft="6px"
                    android:layout_marginRight="6px"
                    android:hint="@string/message_compose_subject_hint"
                    android:inputType="textEmailSubject|textAutoCorrect|textCapSentences|textImeMultiLine"
                    android:imeOptions="actionNext"
                    />
<LinearLayout android:id="@+id/attachments"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:orientation="vertical" />
<View android:layout_width="fill_parent"
                    android:layout_height="1px"
                    android:background="@drawable/divider_horizontal_email" />
</LinearLayout>
<EditText android:id="@+id/message_content"
                android:textColor="?android:attr/textColorSecondaryInverse"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_weight="1.0"
                android:gravity="top"
                android:textAppearance="?android:attr/textAppearanceMedium"
                android:hint="@string/message_compose_body_hint"
                android:inputType="textMultiLine|textAutoCorrect|textCapSentences"
                android:imeOptions="actionDone|flagNoEnterAction"
                />
</LinearLayout>
</ScrollView>
```

☑ android:paddingTop="5dip"，其中 padding 是在父组件 View 的角度指定空间位置，规定其中的内容与这个父 View 边界的距离。margin 则是根据控件自身指定空间位置，设定其他（上、下、左、右）的 View 之间的距离。对应部分代码如下所示。

```
<LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingTop="5dip"
```

```
            android:paddingLeft="4dip"
            android:paddingRight="4dip"
            android:paddingBottom="1dip"
            android:background="@android:drawable/menu_full_frame" >
        <Button
                android:id="@+id/send"
                android:text="@string/send_action"
                android:layout_height="fill_parent"
                android:layout_width="wrap_content"
                android:layout_weight="1" />
        <Button
                android:id="@+id/discard"
                android:text="@string/discard_action"
                android:layout_height="fill_parent"
                android:layout_width="wrap_content"
                android:layout_weight="1" />
        </LinearLayout>
</LinearLayout>
```

第 4 章　移动微博系统

微博是微博客（MicroBlog）的简称，是一个基于用户关系的信息分享、传播以及获取的平台，用户可以通过 Web、WAP 以及各种客户端组建个人社区，以 140 字内的文字更新显示信息，并实现即时分享。在本章的内容中，将详细介绍在 Android 系统中开发微博项目的基本知识。

4.1　微博介绍

知识点讲解：光盘:视频\视频讲解\第 4 章\微博介绍.avi

在当前的互联网时代中，使用博客的用户越来越多，人们常通过博客来抒发情感、编写日记、记录生活中的点点滴滴。为了方便人们的生活，很多智能手机上推出了“移动博客发布器”。其中最早也是最著名的微博是美国的 Twitter，根据相关公开数据，截至 2010 年 1 月份，该产品在全球已经拥有 7500 万注册用户。2009 年 8 月份中国最大的门户网站新浪网推出“新浪微博”内测版，成为门户网站中第一家提供微博服务的网站，微博正式进入中文上网主流人群视野。

（1）微博的特点

微博客草根性更强，且广泛分布在桌面、浏览器、移动终端等多个平台上，有多种商业模式并存，并有形成多个垂直细分领域的可能。但是无论哪种商业模式，都离不开用户体验的特性和基本功能。

（2）手机微博

微博的主要发展运用平台应该是以手机用户为主，微博以计算机为服务器，以手机为平台，把每个手机用户用无线的手机连在一起，让每个手机用户不用使用计算机就可以发表自己的最新信息，并和好友分享自己的快乐。

微博之所以要限定 140 个字符，源于从手机发短信最多的字符就是 140 个。由此可见，微博从诞生之初就同手机应用密不可分，这更是在互联网形态中最大的亮点。微博对互联网的重大意义在于建立手机和互联网应用的无缝连接，培养手机用户使用手机上网的习惯，增强手机端同互联网端的互动，从而使手机用户顺利过渡到无线互联网用户。在目前应用中，手机和微博应用有如下 3 种结合形式。

☑　通过短信和彩信

短信、彩信形式是同移动运营商合作，用户所花费的费用由运营商收取，这种形式覆盖的人群比较广泛，只要能发短信就能更新微博。但对用户来说更新成本太大，并且彩信限制 50KB 大小的弊端严重影响了所发图片的清晰度。最关键的是这个方法只能提供更新，而无法看到其他人的更新，这种单向的信息传输方式大大降低了用户参与性和互动性。

☑　通过 WAP 版网站

各微博网站基本都有自己的 WAP 版，用户可以通过登录 WAP 或通过安装客户端连接到 WAP 版。这种形式只要手机能上网就能连接到微博，可以更新也可以浏览、回复和评论，所需费用就是浏览过程中用的流量费。但目前国内的 GPRS 流量费还相对较高，网速也相对较慢，如果要上传稍大的图片，速度非常慢。

☑　通过手机客户端

手机客户端分如下两种。

- 微博网站开发的基于 WAP 的快捷方式版。

  用户通过客户端直接连接到经过美化和优化的 WAP 版微博网站。这种形式用户行为主要靠主动来实现，也就是用户想起更新和浏览微博时才打开客户端，其实也就相当于在手机端增加了一个微博网站快捷方式，使用操作上的利弊同 WAP 网站也基本相同。

- 利用微博网站提供的 API 开发的第三方客户端。

  这种客户端在国内还比较少，国际上比较有名的是 Twitter 的客户端 Gravity 和 Hesine（和信）。其中 Gravity 是专门为 Twitter 开发的，需要通过主动联网登录，但操作架构和界面经过合理设计，用户体验非常好。和信是国内公司开发的，目前不但支持 Twitter，还支持国内的各主流微博。与其他客户端不同的是，和信的客户端是利用 IP Push 技术提供微博更新和下发通道，不但能够大大提升用户更新微博的速度，而且能将微博消息推送到用户的手机中，用户不用主动登录微博就能实现浏览和互动。和信支持的系统平台比较多，但是缺点是在非智能机上的体验还不是很好。

4.2 微博开发技术介绍

知识点讲解：光盘:视频\视频讲解\第 4 章\微博开发技术介绍.avi

本节将简单介绍在 Android 平台中开发微博系统所需要的基本技术，为读者学习本书后面的知识打下基础。

4.2.1 XML-RPC 技术

开发移动微博系统的关键技术是 RPC（Remote Procedure Call，远程过程调用）。XML-RPC 是一种统一标准的规范，是通过 HTTP 连接的方式运行的，以传送符合 XML-RPC 格式的 request 来调用远程服务器上的某个程序，进而运行博客的功能。现在许多网络服务业者都会以 XML-RPC 的方式提供给软件开发者一个系统介接的管道，让开发者能够根据定义好的方式，以 XML-RPC 的方式来使用该网站的某些功能。当前许多市面中的博客也都支持 XML-RPC 的介接方式。

XML-RPC 的原理是 XML-RPC 工具把传入的参数组合成 XML，然后用通过 HTTP 协议发送给服务器，服务器回复 XML 格式数据，再由专业工具解析给调用者。

在 XML-RPC 标准中，规定 XML 内容的规则如下所示。

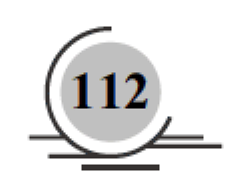
```
<xml version="1.0"?>
<methodCall>
  <methodName>要调用的 method name</methodName>
    <params>
    <param>参数 1</param>
    <param>参数 2</param>
    <param>参数 n</param>
    </params>
</methodCall>
```

Android 本身并不支持 XML-RPC 协议，需要下载相关的工具。读者可以从如下地址下载 XML-RPC。

```
http://code.google.com/p/android-xmlrpc/downloads/list
```

例如，下面的代码演示了用 XML-RPC 协议实现微博客户端的基本过程。

```
package org.xmlrpc;
```



```
import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import org.apache.http.conn.HttpHostConnectException;
import org.xmlrpc.android.XMLRPCClient;
import org.xmlrpc.android.XMLRPCException;
import org.xmlrpc.android.XMLRPCFault;
import org.xmlrpc.android.XMLRPCSerializable;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.util.Log;
import android.widget.EditText;
import android.widget.Toast;
import android.widget.Button;
import android.content.DialogInterface.OnCancelListener;
import android.view.View.OnClickListener;
import android.view.View;

public class TestBlog extends Activity {
        private XMLRPCClient client;
        private URI uri;

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);

                setContentView(R.layout.test_blog);
                Button btn = (Button) findViewById(R.id.send);
                btn.setOnClickListener(new OnClickListener() {
                        public void onClick(View v) {
                                post();
                        }
                });
        }

        void post() {
                String blogid = ((EditText) findViewById(R.id.blogid_edit)).getText()
                                .toString();                    //ID
                String username = ((EditText) findViewById(R.id.username_edit))
                                .getText().toString();          //用户名
                String password = ((EditText) findViewById(R.id.password_edit))
                                .getText().toString();          //密码
                String title = ((EditText) findViewById(R.id.title_edit)).getText()
                                .toString();                    //标题
                String content = ((EditText) findViewById(R.id.content_edit)).getText()
                                .toString();                    //正文
                uri = URI.create("http://blog.csdn.net/" + blogid
                                + "/services/metablogapi.aspx");
                client = new XMLRPCClient(uri);
```

```
            Map<String, Object> structx = new HashMap<String, Object>();
            structx.put("title", title);
            structx.put("description", content);
            Object[ ] params = new Object[ ] { blogid, username, password, structx,
                                true };

            try {
                        client.callEx("metaWeblog.newPost", params);
                        Toast.makeText(this, "OK", 10000).show();
            } catch (XMLRPCException e) {
                        Toast.makeText(this, "ERROR" + e, 10000).show();
            }
    }
```

4.2.2 Meta Weblog API 客户端

Meta Weblog API 是博客园发布的一款功能强大的客户端，其登录地址是 http://www.cnblogs.com/<您的用户名>/services/metaweblog.aspx。Meta Weblog API 支持通过 XML-RPC 的方法在软件中编辑及浏览 Blog，其中最为常用的 API 如下所示。

☑ 发布新文章（metaWeblog.newPost）。
☑ 获取分类（metaWeblog.getCategories）。
☑ 最新文章（metaWeblog.getRecentPosts）。
☑ 新建文章分类（wp.newCategory）。
☑ 上传图片音频或视频（metaWeblog.newMediaObject）。

4.3 在 Android 上开发移动博客发布器

知识点讲解：光盘:视频\视频讲解\第 4 章\在 Android 上开发移动博客发布器.avi

在本实例中实现了一个“移动博客发布器”的功能，以乐多博客为例，演示了如何从手机发布文章到乐多博客上的方法，为读者学习本书后面的知识打下基础。

4.3.1 XML 请求

调用乐多博客的 metaWeblog.newPost 接口实现添加博客文章功能，发出的 XML 请求的内容如下所示。

```
< ?xml version="l.0"?>
<methodCall>
        <methodName>metaWeblog.newPost</methodName>
<params>
                <param><value><string>ID</string></value></param>
                        <param><value><string>账号</string></value></param>
                        <param><value><string>密码</string></value></param>
<param>
 <value>
 <struct>
```

```
<member>
<name>title</name>
        <value><string>文章标题</string></value>
</member>
<member>
<name>descriptiori</name>
        <value><string>内容</string></value>
</member>
</struct>
</value>
</param>
<param><value><boolean>l</boolean></value></param>
</params>
</methodCall>
```

4.3.2　常用接口介绍

并非所有的博客都可以用 GET 或 POST 方式实现 XML-RPC 的 request 交互，有些博客只能以 POST 的方式来传送。建议读者在具体编码之前，先弄清楚服务器接收的 request 是否有特殊限制。在乐多博客的项目中，为开发人员提供了许多交互方法，通过这些方法可以实现多种功能。表 4-1 中列出了几种比较常用的方法。

表 4-1　常用的方法接口

方法名称	参　　数	返　回　值	说　　明
metaWeblog.newPost	博客 ID(string) usename(string) password(string) content publish(boolean)	成功：文章 ID 失败：fault	发布一篇新文章
metaWeblog.editPost	文章 ID(string) usename(string) password(string) content publish(boolean)	成功：true 失败：fault	修改已发布的文章内容
metaWeblog.getPost	文章 ID(string) usename(string) password(string)	成功：文章数组 失败：fault	取得特定文章的信息
metaWeblog.getRecentPosts	博客 ID(string) usename(string) password(string) 返回篇数(int)	成功：文章数组 失败：fault	返回最近发表的文章信息
metaWeblog.deletePost	文章 ID(string) usename(string) password(string)	成功：true 失败：fault	删除已发布的博客文章
mt.getCategoryList	博客 ID(string) usename(string) password(string)	成功：分类数组 失败：fault	取得博客的文章分类信息
mt.getPostCategories	文章 ID(string) usename(string) password(string)	成功：分类数组 失败：fault	返回指定文章的所属类信息

续表

方 法 名 称	参　　数	返　回　值	说　　明
mt.setPostCategories	文章 ID(string) usename(string) password(string) 文章分类（数组）	成功：true 失败：fault	设置指定文章所在的类
mt.supportedMethods		成功：方法数组	取得服务器支持的 XML-RPC 方法列表

4.3.3　具体实现

在本实例中，以 EditText 作为输入博客的相关信息及文章内容的组件。当用户输入完成并单击“发布文章”按钮后，会触发此按钮的 onClick()事件，首先检查输入字段是否是空白，检查无误后，程序先运行 getPostString()，将输入参数转换成符合 XML-RPC 规范的 XML 格式，再调用 sendPost()将 XML 的 request 传送给相对应的博客网址，最后再取得服务器返回的 response，并使用 Dialog 形式显示运行结果。

本实例的具体实现流程如下所示。

（1）编写布局文件 main.xml，主要代码如下所示。

```
<TextView
   android:id="@+id/myText1"
   android:layout_width="wrap_content"
   android:layout_height="23px"
   android:text="@string/str_title1"
   android:textColor="@drawable/black"
   android:layout_x="10px"
   android:layout_y="22px"
>
</TextView>
<TextView
   android:id="@+id/myText2"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/str_title2"
   android:textColor="@drawable/black"
   android:layout_x="10px"
   android:layout_y="62px"
>
</TextView>
<TextView
   android:id="@+id/myText3"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/str_title3"
   android:textColor="@drawable/black"
   android:layout_x="10px"
   android:layout_y="102px"
>
</TextView>
<TextView
   android:id="@+id/myText4"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_title4"
    android:textColor="@drawable/black"
    android:layout_x="10px"
    android:layout_y="142px"
  >
  </TextView>
  <TextView
    android:id="@+id/myText5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/str_title5"
    android:textColor="@drawable/black"
    android:layout_x="10px"
    android:layout_y="182px"
  >
  </TextView>
  <EditText
    android:id="@+id/blogId"
    android:layout_width="100px"
    android:layout_height="40px"
    android:numeric="integer"
    android:layout_x="90px"
    android:layout_y="12px"
  >
  </EditText>
  <EditText
    android:id="@+id/blogAccount"
    android:layout_width="170px"
    android:layout_height="40px"
    android:textSize="16sp"
    android:layout_x="90px"
    android:layout_y="52px"
  >
  </EditText>
  <EditText
    android:id="@+id/blogPwd"
    android:layout_width="170px"
    android:layout_height="40px"
    android:textSize="16sp"
    android:password="true"
    android:layout_x="90px"
    android:layout_y="92px"
  >
  </EditText>
  <EditText
    android:id="@+id/artContent"
    android:layout_width="210px"
    android:layout_height="207px"
    android:textSize="16sp"
```

```
        android:layout_x="90px"
        android:layout_y="172px"
    >
    </EditText>
    <EditText
        android:id="@+id/artTitle"
        android:layout_width="200px"
        android:layout_height="40px"
        android:textSize="16sp"
        android:layout_x="90px"
        android:layout_y="132px"
        android:scrollbars="vertical"
    >
    </EditText>
    <Button
        android:id="@+id/myButton"
        android:layout_width="90px"
        android:layout_height="40px"
        android:text="@string/str_button"
        android:textSize="16sp"
        android:layout_x="120px"
        android:layout_y="382px"
    >
    </Button>
```

（2）编写界面显示文本文件 strings.xml，主要代码如下所示。

```
<resources>
    <string name="hello"></string>
    <string name="app_name"></string>
    <string name="str_title1">ID 号是：</string>
    <string name="str_title2">登录账号：</string>
    <string name="str_title3">登录密码：</string>
    <string name="str_title4">文章标题：</string>
    <string name="str_title5">文章内容：</string>
    <string name="str_button">发布文章</string>
</resources>
```

（3）编写主程序文件 weib.java，主要代码如下所示。

```
public class weib extends Activity
{
    /*变量声明*/
    Button mButton;
    EditText mEdit1;
    EditText mEdit2;
    EditText mEdit3;
    EditText mEdit4;
    EditText mEdit5;
    /*乐多博客 XML-RPC 网址*/
    private String path=
        " http://blog.csdn.net/asdfg343442";
    /* XML-RPC 发布文章的 method name */
    private String method="metaWeblog.newPost";
```

```
@Override
public void onCreate(Bundle savedInstanceState)
{
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  /*初始化对象*/
  mEdit1=(EditText)findViewById(R.id.blogId);
  mEdit2=(EditText)findViewById(R.id.blogAccount);
  mEdit3=(EditText)findViewById(R.id.blogPwd);
  mEdit4=(EditText)findViewById(R.id.artTitle);
  mEdit5=(EditText)findViewById(R.id.artContent);
  mButton=(Button)findViewById(R.id.myButton);
  /*设置发布文章的 onClick 事件*/
  mButton.setOnClickListener(new View.OnClickListener()
  {
    public void onClick(View v)
    {
      /*取得输入的信息*/
      String blogId=mEdit1.getText().toString();
      String account=mEdit2.getText().toString();
      String pwd=mEdit3.getText().toString();
      String title=mEdit4.getText().toString();
      String content=mEdit5.getText().toString();

      if(blogId.equals("")||account.equals("")||pwd.equals("")||
         title.equals("")||content.equals(""))
      {
        showDialog("没有填写内容!");
      }
      else
      {
        /*发送 XML POST 并显示 Response 内容*/
        String outS=getPostString(method,blogId,account,
                                  pwd,title,content);
        String re=sendPost(outS);
        showDialog(re);
      }
    }
  });
}

/*发送 request 至博客的对应网址的 method*/
private String sendPost(String outString)
{
  HttpURLConnection conn=null;
  String result="";
  URL url = null;
  try
  {
    url = new URL(path);
```

```
            conn = (HttpURLConnection)url.openConnection();
            /*允许 Input、Output*/
            conn.setDoInput(true);
            conn.setDoOutput(true);
            /*设置传送的 method=POST*/
            conn.setRequestMethod("POST");
            /*setRequestProperty*/
            conn.setRequestProperty("Content-Type", "text/xml");
            conn.setRequestProperty("Charset", "UTF-8");

            /*送出 request*/
            OutputStreamWriter out =
               new OutputStreamWriter(conn.getOutputStream(), "utf-8");
            out.write(outString);
            out.flush();
            out.close();
            /*解析返回的 XML 内容*/
            result=parseXML(conn.getInputStream());
            conn.disconnect();
         }
         catch(Exception e)
         {
            conn.disconnect();
            e.printStackTrace();
            showDialog(""+e);
         }
         return result;
      }

      /*解析 Response 的 XML 内容的 method*/
      private String parseXML(InputStream is)
      {
         String result="";
         Document doc = null;
         try
         {
            /*将 XML 转换成 Document 对象*/
            DocumentBuilderFactory dbf=
            DocumentBuilderFactory.newInstance();
            DocumentBuilder db=dbf.newDocumentBuilder();
            doc = db.parse(is);
            doc.getDocumentElement().normalize();
            /*检查返回值是否有包含 fault 这个 tag，有就代表发布错误*/
            int fault=doc.getElementsByTagName("fault").getLength();
            if(fault>0)
            {
               result+="发布错误!\n";
               /*取得 faultCode（错误代码）*/
               NodeList nList1=doc.getElementsByTagName("int");
               for (int i = 0; i < nList1.getLength(); ++i)
               {
```

```
            String errCode=nList1.item(i).getChildNodes().item(0)
                                   .getNodeValue();
            result+="错误代码："+errCode+"\n";
          }
          /*取得 faultString（错误信息）*/
          NodeList nList2=doc.getElementsByTagName("string");
          for (int i = 0; i < nList2.getLength(); ++i)
          {
            String errString=nList2.item(i).getChildNodes().item(0)
                                     .getNodeValue();
            result+="错误信息："+errString+"\n";
          }
        }
        else
        {
          /*发布成功，取得文章编号*/
          NodeList nList=doc.getElementsByTagName("string");
          for (int i = 0; i < nList.getLength(); ++i)
          {
            String artId=nList.item(i).getChildNodes().item(0)
                                 .getNodeValue();
            result+="发布成功!!文章编号「"+artId+"」";
          }
        }
      }
      catch (Exception ioe)
      {
        showDialog(""+ioe);
      }
      return result;
    }

    /*一组要发送的 XML 内容的 method*/
    private String getPostString(String method,String blogId,
        String account,String pwd,String title,String content)
    {
      String s="";
      s+="<methodCall>";
      s+="<methodName>"+method+"</methodName>";
      s+="<params>";
      s+="<param><value><string>"+blogId+"</string></value></param>";
      s+="<param><value><string>"+account+"</string></value></param>";
      s+="<param><value><string>"+pwd+"</string></value></param>";
      s+="<param><value><struct>";
      s+="<member><name>title</name>" +
         "<value><string>"+title+"</string></value></member>";
      s+="<member><name>description</name>" +
         "<value><string>"+content+"</string></value></member>";
      s+="</struct></value></param>";
      s+="<param><value><boolean>1</boolean></value></param>";
      s+="</params>";
```

```
        s+="</methodCall>";

        return s;
    }

    /*跳出 Dialog 的 method*/
    private void showDialog(String mess)
    {
        new AlertDialog.Builder(weib.this).setTitle("Message")
        .setMessage(mess)
        .setNegativeButton("确定", new DialogInterface.OnClickListener()
        {
            public void onClick(DialogInterface dialog, int which)
            {
            }
        })
        .show();
    }
}
```

执行后的效果如图 4-1 所示，只要拥有乐多的账号，就可以在手机上发送移动博客。



图 4-1　执行效果

4.4　分析腾讯 Android 版微博 API

知识点讲解：光盘:视频\视频讲解\第 4 章\分析腾讯 Android 版微博 API.avi

对 Android 学习者来说，个人独立开发微博系统的难度比较大。在当前市面中有很多著名微博系统的开源代码，开发人员只需利用所提供的 API 接口，就可以方便地开发出 Android 版的移动微博系统。当今市面中著名的 Android 版的移动微博系统有新浪微博和腾讯微博。在本节的内容中，将首先讲解腾讯微博系统的 API 接口。

4.4.1　源码和 jar 包下载

因为当前腾讯微博提供的 Java（Android）SDK 功能过弱，所以特意集成了一个 Java SDK 包，此包适用于 Android 系统，包中包含了腾讯微博目前提供的 95%的 API，主要特点如下所示。

☑ 用法简单：微博、评论、转发、私信同一个实体类。
☑ 方便扩展：可以根据需要修改源代码或是继承 QqTSdkService 类，为了后续依然能升级版本建议采用继承的方式。

在压缩包 Java SDK 中，QqTAndroidSdk-1.0.0.jar 是 SDK 的主代码，其中 QqTSdkServiceImpl 包含了所有接口的实现。各个包的具体说明如下所示。

☑ jar 包地址：QqTAndroidSdk-1.0.0.jar。
☑ Google Code 源码地址：http://code.google.com/p/qq-t-java-sdk/source/browse/。
☑ Github 源码地址：https://github.com/Trinea/qq-t-java-sdk。
☑ 压缩包 JavaCommon-1.0.0.jar：是 QqTAndroidSdk 依赖的公用处理包，包含字符串、list、数组、map、json 工具类等。
☑ jar 包地址：已经包含在 QqTAndroidSdk-1.0.0.jar 中。
☑ Google Code 源码地址：http://code.google.com/p/trinea-java-common/source/browse/。
☑ Github 源码地址：https://github.com/Trinea/JavaCommon。

4.4.2　具体使用

在具体使用之前，读者请先参考腾讯微博的 API 文档说明，地址是 http://wiki.open.t.qq.com/ index.php/ API%E6%96%87%E6%A1%A3，如图 4-2 所示。



图 4-2　腾讯微博的 API 文档页面

在编码使用 API 接口时，都需要先新建 QqTSdkService 类对象并进行初始化工作。例如下面的初始化代码。

```
/**
 * 分别设置应用的 key、secret（腾讯提供）。用户的 accesstoken 和 tokenSecret（OAuth 获取）
 * 请用自己的相应字符串替换，否则无法成功发送和获取数据
 **/
QqTAppAndToken qqTAppAndToken = new QqTAppAndToken();
qqTAppAndToken.setAppKey("***");                //***用应用 key 替换
qqTAppAndToken.setAppSecret("***");             //***用应用 secret 替换
qqTAppAndToken.setAccessToken("***");           //***用用户 accesstoken 替换
qqTAppAndToken.setTokenSecret("***");           //***用用户 tokenSecret 替换

/** 新建 QqTSdkService 对象，并设置应用信息和用户访问信息 **/
QqTSdkService qqTSdkService = new QqTSdkServiceImpl();
qqTSdkService.setQqTAppAndToken(qqTAppAndToken);
```

接下来开始对接口进行详细介绍，并讲解使用 QqTAndroidSdk-1.0.0.jar 中的 API 的方法。腾讯微博中的接口主要分成下面的几大类。

1. 时间线（微博列表）

这里的 20 个接口包含了腾讯微博的如下 4 部分 API。

（1）时间线中的除 statuses/ht_timeline_ext（话题时间线）以外的 15 个 API。

（2）私信相关中的收件箱、发件箱两个 API。

（3）数据收藏中收藏的微博列表和获取已订阅话题列表的两个 API。

（4）微博相关中获取单条微博的转发或点评列表 API。

以获取首页信息为例，示例代码如下所示。

```
QqTTimelinePara qqTTimelinePara = new QqTTimelinePara();
/** 设置分页标识 **/
qqTTimelinePara.setPageFlag(0);
/** 设置起始时间 **/
qqTTimelinePara.setPageTime(0);
/** 每次请求记录的条数 **/
qqTTimelinePara.setPageReqNum(QqTConstant.VALUE_PAGE_REQ_NUM);
/** 可以设置拉取类型，可取值 QqTConstant 中 VALUE_STATUS_TYPE_TL_... **/
qqTTimelinePara.setStatusType(QqTConstant.VALUE_STATUS_TYPE_TL_ALL);
/** 可以设置微博内容类型，可取值 QqTConstant 中 VALUE_CONTENT_TYPE_TL... **/
qqTTimelinePara.setContentType(QqTConstant.VALUE_CONTENT_TYPE_TL_ALL);
List<QqTStatus> qqTStatusList = qqTSdkService.getHomeTL(qqTTimelinePara);
assertTrue(qqTStatusList != null);
```

这样 qqTStatusList 就保存了首页的 20 条数据，可以自行设置不同的类型参数。如果想获取更多时间线数据，请读者参考腾讯微博 Java（android）SDK 时间线 API 的详细介绍。

2. 新增微博 API

在本书成稿时，腾讯微博新增加了 8 个 API。

（1）微博相关中发表一条微博、转播一条微博、回复一条微博、发表一条带图片微博、点评一条微博、发表音乐微博、发表视频微博、发表心情帖子。在 API 中发表一条微博和发表一条带图片微博合二为一。

（2）私信相关中的发私信，以新增一条微博为例，演示代码如下。

```
qqTSdkService.addStatus("第一条状态哦", null);
```

其第一个参数为状态内容，第二个参数为图片地址，不传图片为空即可。或者在如下复杂代码中，status 可以设置其他地理位置信息。



```
QqTStatusInfoPara status = new QqTStatusInfoPara();
status.setStatusContent("发表一条带图片微博啦");
/** 发表带图微博，设置图片路径 **/
status.setImageFilePath("/mnt/sdcard/DCIM/Camera/IMAG2150.jpg");
assertTrue(qqTSdkService.addStatus(status, qqTAppAndToken));
```

这 6 个接口包含了腾讯微博 3 部分 API。

3. 操作一条微博

（1）在微博相关中删除一条微博 API。

（2）在私信相关中删除私信 API。

（3）在数据收藏中收藏微博、取消收藏微博、订阅话题、取消订阅话题 4 个 API。以收藏一条微博为例，示例代码如下：

```
qqTSdkService.collect(12121);
```

其中参数为微博 ID。

4. 关系链列表（用户列表）

关系链列表这 10 个接口分别包含了腾讯微博关系链相关中的互听关系链列表（对某个用户而言，既是他的听众又被他收听）、其他账号听众列表、其他账号收听的用户的列表、其他账号特别收听的用户的列表、黑名单列表、我的听众列表、我的听众列表（只包含名字）、我收听的人列表、我收听的人列表（只包含名字）、我的特别收听列表。

以获取自己的收听用户为例，示例代码如下所示。

```
QqTUserRelationPara qqTUserRelationPara = new QqTUserRelationPara();
qqTUserRelationPara.setReqNumber(QqTConstant.VALUE_PAGE_REQ_NUM);
qqTUserRelationPara.setStartIndex(0);
List<QqTUser> qqTUserList = qqTSdkService.getSelfInterested(qqTUserRelationPara);
```

5. 用户建立关系

用户建立关系包含 6 个接口，这 6 个接口分别包含了腾讯微博关系链相关中的收听某个用户、取消收听某个用户、特别收听某个用户、取消特别收听某个用户、添加某个用户到黑名单、从黑名单中删除某个用户。

以关注某些用户为例，示例代码如下所示。

```
qqTSdkService.interestedInOther("wenzhang,li_nian,mayili007", null)
```

6. 账户相关

账户相关包含 7 个接口，这 7 个接口分别包含了腾讯微博账户相关中的获取自己的详细资料、更新用户信息、更新用户头像信息、更新用户教育信息、获取其他人资料、获取一批人的简单资料、验证账户是否合法（是否注册微博）。（除获取心情微博 API 外。）

以获取自己的资料为例，示例代码如下所示。

```
QqTUser qqTUser = qqTSdkService.getSelfInfo();
```

7. 搜索相关

搜索相关包含了腾讯微博搜索相关中的搜索用户、搜索微博、通过标签搜索用户共 3 个 API。以搜索微博为例，示例代码如下所示。

```
public void testSearchStatus() {
```

```
        QqTSearchPara qqTSearchPara = new QqTSearchPara();
        qqTSearchPara.setKeyword("iphone");
        qqTSearchPara.setPage(1);
        qqTSearchPara.setPageSize(QqTConstant.VALUE_PAGE_REQ_NUM);
        List<QqTStatus> qqTStatusList = qqTSdkService.searchStatus(qqTSearchPara);
        assertTrue(qqTStatusList != null);
}
```

8. 热度趋势相关

热度趋势相关包含了腾讯微博热度趋势中的话题热榜、转播热榜用户共两个 API。以话题热榜为例，示例代码如下所示。

```
public void testGetHotTopics() {
        QqTHotStatusPara qqTHotStatusPara = new QqTHotStatusPara();
        qqTHotStatusPara.setReqNum(QqTConstant.VALUE_PAGE_REQ_NUM);
        qqTHotStatusPara.setLastPosition(0);
        /**
         * 1 话题名，2 搜索关键字，3 两种类型都有
         **/
        qqTHotStatusPara.setType(Integer.toString(1));
        List<QqTTopicSimple> hotTopicsList = qqTSdkService.getHotTopics(qqTHotStatusPara);
        assertTrue(hotTopicsList != null);
}
```

9. 数据更新

数据更新为腾讯微博数据更新相关中的查看数据更新条数 API，示例代码如下所示。

```
public void testGetUpdateInfoNum() {
        /** 设置 clearType，对应 QqTConstant.VALUE_CLEAR_TYPE_... **/
        QqTUpdateNumInfo qqTUpdateNumInfo = qqTSdkService.getUpdateInfoNum(true, QqTConstant.VALUE_
CLEAR_TYPE_HOME_PAGE);
        assertTrue(qqTUpdateNumInfo != null);
}
```

10. 发起话题

发起话题有两个接口，这两个接口是为腾讯微博中的话题应用服务的，可以根据话题名称查询话题 ID 和根据话题 ID 获取话题相关信息。示例代码如下所示。

```
public void testGetTopicInfoByIds() {
        /**先得到话题 ID**/
        Map<String, String> topicIdAndName = qqTSdkService.getTopicIdByNames("袁莉闪婚,美汁源下架,
iphone");
        if (topicIdAndName != null) {
                /** 话题 ID 列表，以逗号分隔 **/
                List<QqTStatus> qqtStatusList = qqTSdkService.getTopicInfoByIds(ListUtils.join(new ArrayList<String>
(topicIdAndName.keySet())));
                assertTrue(qqtStatusList != null);
        } else {
                assertTrue(false);
        }
}
```

11. 标签相关

标签相关有两个接口，这两个接口为腾讯微博标签相关中的添加标签和删除标签，示例代码如下所示。

```
public void testDeleteTag() {
    /** 删除自己的 tag，先获取自己的资料，从中取得 tag id **/
    QqTUser qqTUser = qqTSdkService.getSelfInfo();
    if (qqTUser != null && qqTUser.getTagMap() != null && qqTUser.getTagMap().size() > 0) {
        /** 删除 tag **/
        for (Map.Entry<String, String> tag : qqTUser.getTagMap().entrySet()) {
            qqTSdkService.deleteTag(tag.getKey());
        }
    } else {
        assertTrue(false);
    }
}
```

4.5　详解新浪 Android 版微博 API

知识点讲解：光盘:视频\视频讲解\第 4 章\详解新浪 Android 版微博 API.avi

新浪微博是国内最早推出微博应用的行业站点，为了帮助 Android 程序员使用新浪微博中的应用，特意提供了开源 API 供大家参考。读者要想了解在 Android 平台使用新浪微博的知识，可以登录 http://open.weibo.com/wiki/%E9%A6%96%E9%A1%B5 获取详细资料，并且可在网页中获取开源代码。

在 Android 使用新浪微博的开发平台 API 的基本步骤如下所示。

1. 通过官方网址下载 SDK

当前的最新版本是 Weibo4Android，下载页面的地址是 http://code.google.com/p/weibo4j/downloads/detail?name=weibo4android-1.2.1.zip，此页面的界面效果如图 4-3 所示。



图 4-3　下载 SDK 页面

2. 认证

在 SDK 中有完整的如何通过 OAuth 认证的演示实例，认证和使用流程大概如下。

（1）在/weibo4android/src/weibo4android/Weibo.java 中设置 App Key 和 App Secret（在官方网站新建应用可获得），代码如下所示。

```
public static String CONSUMER_KEY = "2664209963";
public static String CONSUMER_SECRET = "b428615797a5d676d428cd146c040399";
```

（2）在/weibo4android/examples/weibo4android/androidexamples/AndroidExample.java 中，将 App Key 和 App Secret 设置进系统类中。

```
System.setProperty("weibo4j.oauth.consumerKey", Weibo.CONSUMER_KEY);
System.setProperty("weibo4j.oauth.consumerSecret", Weibo.CONSUMER_SECRET);
```

（3）通过 HTTP POST 方式向服务提供方请求获得 RequestToken。

```
RequestToken   requestToken =weibo.getOAuthRequestToken("weibo4android://OAuthActivity");
```

（4）将用户引导至授权页面。

```
Uri uri = Uri.parse(requestToken.getAuthenticationURL()+ "&display=mobile");
startActivity(new Intent(Intent.ACTION_VIEW, uri));
```

（5）授权页面要求用户输入用户名和密码，授权完成后，服务提供方会通过回调 URI 将用户引导回客户端页面 OAuthActivity。

```
<activity android:name=".OAuthActivity">
  <intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data android:scheme="weibo4android" android:host="OAuthActivity" />
    </intent-filter>
</activity>
```

（6）客户端根据临时令牌和用户授权码从服务提供方获取访问令牌（Access Token）。

```
Uri uri=this.getIntent().getData();
RequestToken requestToken= OAuthConstant.getInstance().getRequestToken();
AccessToken accessToken=requestToken.getAccessToken(uri.getQueryParameter("oauth_verifier"));
```

（7）获得访问令牌后便可使用 API 接口获得和操作用户数据。

```
Weibo weibo=OAuthConstant.getInstance().getWeibo();
weibo.setToken(OAuthConstant.getInstance().getToken(), OAuthConstant.getInstance().getTokenSecret());
String[ ] args = new String[2];
args[0]=OAuthConstant.getInstance().getToken();
args[1]=OAuthConstant.getInstance().getTokenSecret();
try {
     GetFollowers.main(args);//返回用户关注对象列表，并返回最新微博文章
   } catch (Exception e) {
     e.printStackTrace();
}
```

在上述步骤中，weibo4android 是 XML 文件中定义的索引名，在上面步骤（5）中的 XML 代码中，<data android:scheme="weibo4android" android:host="OAuthActivity" />部分的索引名是自定义的，只要与 Java 代码中的 URL 匹配即可。

在本书接下来的内容中，将不再剖析 Android 版新浪微博的实现源码，而是以此为基础，讲解二次扩展开发的基本知识。

4.5.1　新浪微博图片缩放的开发实例

在 Android 开发过程中，有时会用到图片缩放效果，即单击图片时显示缩放按钮，过一会消失。接下来将根据新浪微博的图片缩放原理编写演示代码以供参考。

（1）UI 布局文件的演示代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android：orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/layout1"
        >

    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:id="@+id/rl"
            >

        <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="19"
            android:scrollbars="none"
            android:fadingEdge="vertical"
            android:layout_gravity="center"
            android:gravity="center"
            >

        <HorizontalScrollView
            android:layout_height="fill_parent"
            android:layout_width="fill_parent"
            android:scrollbars="none"
            android:layout_gravity="center"
            android:gravity="center"
            android:id="@+id/hs"

            >
            <LinearLayout
                Android:orientation="horizontal"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:id="@+id/layoutImage"
                android:layout_gravity="center"
                android:gravity="center"
                >
                <ImageView
                    android:layout_gravity="center"
                    android:gravity="center"
                    android:id="@+id/myImageView"
```

```
                                        android:layout_width="fill_parent"
                                        android:layout_height="fill_parent"
                                        android:layout_weight="19"
                                        android:paddingTop="5dip"
                                        android:paddingBottom="5dip"

                                        />
                        </LinearLayout>
                </HorizontalScrollView >
                </ScrollView>

                        <ZoomControls android:id="@+id/zoomcontrol"
                        android:layout_width="wrap_content" android:layout_height="wrap_content"
                        android:layout_centerHorizontal="true"
                        android:layout_alignParentBottom="true"
              >
                        </ZoomControls>
                </RelativeLayout>

    </FrameLayout>
```

（2）用 Java 编写主程序代码，代码如下所示。

```
package com.Johnson.image.zoom;
import android.app.Activity;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.DialogInterface.OnKeyListener;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.os.Bundle;
import android.os.Handler;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.ZoomControls;
public class MainActivity extends Activity {
        /** Called when the activity is first created*/
  private final int LOADING_IMAGE = 1;
  public static String KEY_IMAGEURI = "ImageUri";
  private ZoomControls zoom;
  private ImageView mImageView;
  private LinearLayout layoutImage;
  private int displayWidth;
  private int displayHeight;
```

```
/*图片资源*/
private Bitmap bmp;
/*宽的缩放比例*/
private float scaleWidth = 1;
/*高的缩放比例*/
private float scaleHeight = 1;
/*用来计数放大+1，缩小-1*/
private int zoomNumber=0;
/*单击屏幕显示缩放按钮，3 秒消失*/
private int showTime=3000;
RelativeLayout rl;
Handler mHandler = new Handler();
private Runnable task = new Runnable() {
   public void run() {
      zoom.setVisibility(View.INVISIBLE);
   }
};
         @Override
         public void onCreate(Bundle savedInstanceState) {
                  super.onCreate(savedInstanceState);
                  setContentView(R.layout.main);
   //showDialog(LOADING_IMAGE);
   //若图片是从网络上获取的，需要加入滚动条
         bmp=BitmapFactory.decodeResource(getResources(), R.drawable.image);
   //removeDialog(LOADING_IMAGE);
          initZoom();
}
   @Override
   protected Dialog onCreateDialog(int id) {
      switch (id) {
      case LOADING_IMAGE: {
         final ProgressDialog dialog = new ProgressDialog(this);
         dialog.setOnKeyListener(new OnKeyListener() {
            @Override
            public boolean onKey(DialogInterface dialog, int keyCode,
                  KeyEvent event) {
               if (keyCode == KeyEvent.KEYCODE_BACK) {
                  finish();
               }
               return false;
            }
         });
         dialog.setMessage("正在加载图片请稍后...");
         dialog.setIndeterminate(true);
         dialog.setCancelable(true);
         return dialog;
      }
      }
      return null;
   }
   public void initZoom() {
```

```
/*取得屏幕分辨率大小*/
DisplayMetrics dm = new DisplayMetrics();
getWindowManager().getDefaultDisplay().getMetrics(dm);
displayWidth = dm.widthPixels;
displayHeight = dm.heightPixels;
mImageView = (ImageView) findViewById(R.id.myImageView);
mImageView.setImageBitmap(bmp);
layoutImage = (LinearLayout) findViewById(R.id.layoutImage);
mImageView.setOnClickListener(new OnClickListener() {

  @Override
  public void onClick(View v) {
    // TODO Auto-generated method stub
                                    /**
                                     * 在图片上和整个 view 上同时添加单击监听捕捉屏幕
                                     * 单击事件，来显示放大和缩小按钮
                                     * */
    zoom.setVisibility(View.VISIBLE);
    mHandler.removeCallbacks(task);
    mHandler.postDelayed(task, showTime);
  }
});
layoutImage.setOnClickListener(new OnClickListener() {

  @Override
  public void onClick(View v) {
    // TODO Auto-generated method stub

    zoom.setVisibility(View.VISIBLE);
    mHandler.removeCallbacks(task);
    mHandler.postDelayed(task, showTime);
  }
});

zoom = (ZoomControls) findViewById(R.id.zoomcontrol);
zoom.setIsZoomInEnabled(true);
zoom.setIsZoomOutEnabled(true);
//图片放大
zoom.setOnZoomInClickListener(new OnClickListener() {
  public void onClick(View v) {
    big();
  }
});
//图片减小
zoom.setOnZoomOutClickListener(new OnClickListener() {

  public void onClick(View v) {
    small();
  }
```

```
    });
    zoom.setVisibility(View.VISIBLE);
    mHandler.postDelayed(task, showTime);

}

@Override
public boolean onTouchEvent(MotionEvent event) {
    // TODO Auto-generated method stub
        /**
                * 在图片上和整个 view 上同时添加单击监听捕捉屏幕
                * 单击事件，来显示放大和缩小按钮
                * */
    zoom.setVisibility(View.VISIBLE);
    mHandler.removeCallbacks(task);
    mHandler.postDelayed(task, showTime);
    return false;
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    // TODO Auto-generated method stub
    super.onKeyDown(keyCode, event);

    return true;
}

/*图片缩小的 method*/
private void small() {
    --zoomNumber;
    int bmpWidth = bmp.getWidth();
    int bmpHeight = bmp.getHeight();

    Log.i("","bmpWidth = " + bmpWidth + ", bmpHeight = " + bmpHeight);

    /*设置图片缩小的比例*/
    double scale = 0.8;
    /*计算出这次要缩小的比例*/
    scaleWidth = (float) (scaleWidth * scale);
    scaleHeight = (float) (scaleHeight * scale);
    /*产生 resize 后的 Bitmap 对象*/
    Matrix matrix = new Matrix();
    matrix.postScale(scaleWidth, scaleHeight);
    Bitmap resizeBmp = Bitmap.createBitmap(bmp, 0, 0, bmpWidth, bmpHeight,
        matrix, true);
    mImageView.setImageBitmap(resizeBmp);

    /*限制缩小尺寸*/
    if ((scaleWidth * scale * bmpWidth < bmpWidth / 4
        || scaleHeight * scale * bmpHeight > bmpWidth /4
        || scaleWidth * scale * bmpWidth > displayWidth / 5
```

```
        || scaleHeight * scale * bmpHeight > displayHeight / 5)&&(zoomNumber==-1) ){

    zoom.setIsZoomOutEnabled(false);

    } else {

    zoom.setIsZoomOutEnabled(true);

    }

    zoom.setIsZoomInEnabled(true);
    System.gc();
  }

  /*图片放大的 method*/
  private void big() {
    ++zoomNumber;
    int bmpWidth = bmp.getWidth();
    int bmpHeight = bmp.getHeight();

    /*设置图片放大的比例*/
    double scale = 1.25;
    /*计算这次要放大的比例*/
    scaleWidth = (float) (scaleWidth * scale);
    scaleHeight = (float) (scaleHeight * scale);
    /*产生 resize 后的 Bitmap 对象*/
    Matrix matrix = new Matrix();
    matrix.postScale(scaleWidth, scaleHeight);
    Bitmap resizeBmp = Bitmap.createBitmap(bmp, 0, 0, bmpWidth, bmpHeight,
        matrix, true);
    mImageView.setImageBitmap(resizeBmp);
    /*限制放大尺寸*/
    if (scaleWidth * scale * bmpWidth > bmpWidth * 4
        || scaleHeight * scale * bmpHeight > bmpWidth * 4
        || scaleWidth * scale * bmpWidth > displayWidth * 5
        || scaleHeight * scale * bmpHeight > displayHeight * 5) {

      zoom.setIsZoomInEnabled(false);

    } else {

      zoom.setIsZoomInEnabled(true);

    }

    zoom.setIsZoomOutEnabled(true);

  System.gc();
  }

}
```

4.5.2　添加分享到新浪微博

现在很多平台都开放了，并且提供了相应的接口。在过去浏览论坛或者博客时，每一个论坛或博客都需要一个专用账号，但是现在会发现很多网站都有一个“用新浪微博登录”、“用 QQ 账号登录”之类的字样。经过授权以后可以用新浪或腾讯的账号登录到不同论坛或者博客，这给用户带来了很多方便。

最近开发的应用有的涉及到分享功能，Android 系统有内置的分享功能，但是内置的分享只有安装该应用时才会被显示在列表中，下面是 Android 系统内置的分享，如图 4-4 所示。



图 4-4　Android 系统内置的分享

选择图 4-4 中的“分享”选项后可以看到新浪微博，这个是笔者自己添加的。如果有安装“新浪微博”移动端，就用系统自己的分享。如果没有安装该应用，则需自行添加分享到“新浪微博”的功能。下面先看看这个列表是怎么加载出来的。

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
ShareAdapter mAdapter = new ShareAdapter(mContext, intent);
//对话框的适配器
public class ShareAdapter extends BaseAdapter {
	private final static String PACKAGENAME = "com.sina.weibo";
	private Context mContext;
	private PackageManager mPackageManager;
	private Intent mIntent;
	private LayoutInflater mInflater;
	private List<ResolveInfo> mList;
	private List<DisplayResolveInfo> mDisplayResolveInfoList;
	public ShareAdapter(Context context, Intent intent) {
		mContext = context;
		mPackageManager = mContext.getPackageManager();
		mIntent = new Intent(intent);
		mInflater = (LayoutInflater)mContext.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
		mList = mContext.getPackageManager().queryIntentActivities(intent,
				PackageManager.MATCH_DEFAULT_ONLY);
		//排序
		ResolveInfo.DisplayNameComparator comparator = new ResolveInfo.DisplayNameComparator(
				mPackageManager);
```

```
        Collections.sort(mList, comparator);
        mDisplayResolveInfoList = new ArrayList<DisplayResolveInfo>();
        if (mList == null || mList.isEmpty()) {
            mList = new ArrayList<ResolveInfo>();
        }
        final int N = mList.size();
        for (int i = 0; i < N; i++) {
            ResolveInfo ri = mList.get(i);
            CharSequence label = ri.loadLabel(mPackageManager);
            DisplayResolveInfo d = new DisplayResolveInfo(ri, null, null, label, null);
            mDisplayResolveInfoList.add(d);
        }
        //考虑是否已安装新浪微博，如果没有则自行添加
        if(!isInstallApplication(mContext, PACKAGENAME)){
            Intent i = new Intent(mContext, ShareActivity.class);
            Drawable d = mContext.getResources().getDrawable(R.drawable.sina);
            CharSequence label = mContext.getString(R.string.about_sina_weibo);
            DisplayResolveInfo dr = new DisplayResolveInfo(null, i, null, label, d);
            mDisplayResolveInfoList.add(0, dr);
        }
    }
    @Override
    public int getCount() {
        return mDisplayResolveInfoList.size();
    }

    @Override
    public Object getItem(int position) {
        return mDisplayResolveInfoList.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View item;
        if(convertView == null) {
            item = mInflater.inflate(R.layout.share_item, null);
        } else {
            item = convertView;
        }
        DisplayResolveInfo info = mDisplayResolveInfoList.get(position);

        ImageView i = (ImageView) item.findViewById(R.id.share_item_icon);
        if(info.mDrawable == null){
            i.setImageDrawable(info.mResoleInfo.loadIcon(mPackageManager));
        }else{
            i.setImageDrawable(info.mDrawable);
        }
```

```
            TextView t = (TextView) item.findViewById(R.id.share_item_text);
            t.setText(info.mLabel);
            return item;
      }

      public ResolveInfo getResolveInfo(int index){
            if(mDisplayResolveInfoList == null){
                  return null;
            }
            DisplayResolveInfo d = mDisplayResolveInfoList.get(index);
            if(d.mResoleInfo == null){
                  return null;
            }
            return d.mResoleInfo;
      }

      //返回跳转 intent
      public Intent getIntentForPosition(int index) {
            if(mDisplayResolveInfoList == null){
                  return null;
            }
            DisplayResolveInfo d = mDisplayResolveInfoList.get(index);
            Intent i = new Intent(d.mIntent == null ? mIntent : d.mIntent);
            i.addFlags(Intent.FLAG_ACTIVITY_FORWARD_RESULT | Intent.FLAG_ACTIVITY_PREVIOUS_IS_TOP);
            if(d.mResoleInfo != null){
                  ActivityInfo a = d.mResoleInfo.activityInfo;
                  i.setComponent(new ComponentName(a.applicationInfo.packageName, a.name));
            }
            return i;
      }

      //检查是否安装该 App
      boolean isInstallApplication(Context context, String packageName){
            try {
                  mPackageManager
                              .getApplicationInfo(packageName,
PackageManager.GET_UNINSTALLED_PACKAGES);
                  return true;
            } catch (NameNotFoundException e) {
                  return false;
            }

      }

      /**
       * 打包数据 VO
       * @author Administrator
       */
      class DisplayResolveInfo {
            private Intent mIntent;
```

```
            private ResolveInfo mResoleInfo;
            private CharSequence mLabel;
            private Drawable mDrawable;

            DisplayResolveInfo(ResolveInfo resolveInfo, Intent intent,
                        CharSequence info, CharSequence label, Drawable d) {
                this.mIntent = intent;
                this.mResoleInfo = resolveInfo;
                this.mLabel = label;
                this.mDrawable = d;
            }
      }
}
```

以上代码加载弹出框的数据适配器，如果系统有安装则直接读取系统的分享，没有则添加。当单击分享微博时需要一列的验证和授权，此处采用的机制是先获取 requestToken，然后通过 requestToken 获取 AccessToken，然后才可以分享微博。

接下来在单击“分享到微博”按钮时进行用户认证，首先说明此处的认证是读取新浪提供的页面，下面是部分代码。

```
Weibo weibo = new Weibo();
RequestToken requestToken = weibo.getOAuthRequestToken("yunmai://ShareActivity");<span style="color:
#e53333;"> //与配置中对应</span> Log.i(TAG, "token:" + requestToken.getToken() + ",tokenSecret:" +
requestToken.getTokenSecret());
OAuthConstant.getInstance().setRequestToken(requestToken);
Uri uri = Uri.parse(requestToken.getAuthenticationURL() + "&display=mobile");
url = uri.toString();
```

上面的地址 URI 就是请求新浪提供的登录界面的地址，此时会涉及 webview 的使用。Android 中 webview 其实就是一个小型浏览器，功能很强大，强大到可以执行脚本。有了地址即可通过 webview.loadurl（url）请求登录界面。也有很多网友可能会想自己设计一个登录界面，但是新浪官方有说明通过 getXauth-AccessToken 方式认证是可以自行设计登录界面的，其他认证方式是不能够自行设计的。单击“授权”按钮时需要跳转到自己的 Activity 中，此处的配置是需要在 androidmanifest.xml 中配置的，例如设置跳转到 shareactivity.java。

```
<activity
            android:name="cn.yunmai.cclauncher.ShareActivity"
            android:screenOrientation="portrait" >
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />
                <data android:host="ShareActivity" android:scheme="yunmai" />
            </intent-filter>
</activity>
```

在此需要注意，<data>标签中的内容需要和显示在授权窗口中的 weibo.getOAuthRequestToken ("yunmai://ShareActivity")相对应。当授权完成之后就会跳转到自己定义的 activity，授权完成之后就会将微博发送到跳转之后的 Activity。在单击“发送”按钮时，要获取刚才授权成功的 RequestToken，然后再获取 accessToken，最后发送微博。在此需要注意的是，RequestToken 只需获取一次，然后保存每次都需要使用的口令 accessToken。单击“发送”按钮的操作代码如下。

```
Uri uri = this.getIntent().getData();
```

```
RequestToken requestToken = OAuthConstant.getInstance().getRequestToken();
AccessToken accessToken = requestToken.getAccessToken(uri.getQueryParameter("oauth_verifier"));
saveAccessToken(accessToken);//保存 accessToken
Log.i(TAG, "oauth_verifier:" + uri.getQueryParameter("oauth_verifier") +
                        ",Token" + accessToken.getToken() + ",TokenSecret:" + accessToken.getTokenSecret());
OAuthConstant.getInstance().setAccessToken(accessToken);
```

此处所执行的操作是获取授权之后的 accessToken，然后发送微博。

```
Weibo weibo = OAuthConstant.getInstance().getWeibo();
weibo.setToken(OAuthConstant.getInstance().getToken(), OAuthConstant.getInstance().getTokenSecret());
Status s = weibo.updateStatus(mEdit.getText().toString());
```

status 返回了一些详细的信息，有发送时间和用户 ID 等，这样就完成了分享功能。

4.5.3　通过 JSON 对象获取登录新浪微博

可以引用新浪开发包中的各种类，在 Android 中通过 JSON 对象的方式登录新浪微博。在下面的代码中，1 代表登录成功，0 代表登录失败，并通过方法 verifyCredentials()请求新浪微博服务器返回 JSON 对象。

```
package com.sfc.ui;

import java.util.ArrayList;
import java.util.List;

import com.sfc.ui.adapter.LoginListAdapter;

import weibo4j.User;                          //这是新浪开发包中的实体类
import weibo4j.Weibo;                         //这是新浪开发包中的类
import weibo4j.WeiboException;                //这是新浪开发包中的类

import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Toast;

public class LoginActivity extends Activity implements Runnable {
 private Button loginButton;
 private ListView listView;
 private ProgressDialog loginDialog;
 private Thread loginThread;
 private Handler handler;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
```

```
  setContentView(R.layout.login);
  loginButton = (Button)findViewById(R.id.loginButton);
  List<String> list = new ArrayList<String>();
  list.add("随便看看");
  list.add("推荐用户");
  list.add("热门转发");
  listView = (ListView)findViewById(R.id.listView);
  loginThread = new Thread(this);

  handler = new Handler(){
              //1 代表登录成功，0 代表登录失败
           public void handleMessage(Message msg) {
            loginDialog.cancel();
                switch (msg.what) {
              case 1:
               Toast.makeText(LoginActivity.this, "登录成功  ", 3000).show();
                  break;
              case 0:
               Toast.makeText(LoginActivity.this, "登录失败", 3000).show();
                  break;
              }
           };
  };
  listView.setAdapter(new LoginListAdapter(this,list));
  loginButton.setOnClickListener(new OnClickListener(){
       public void onClick(View v) {
       loginDialog = new ProgressDialog(LoginActivity.this);
       loginDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
       loginDialog.setMessage("登录服务器");
       loginDialog.show();
       loginThread.start();
   }
  });
 }
 public void run() {
    Log.e("loginThread","start");
    Weibo weibo = new Weibo("XXX@sina.com","XXX"); //新浪微博用户名和密码
    weibo.setHttpConnectionTimeout(5000);
    Message msa = new Message();
    try {
       User user = weibo.verifyCredentials();            //该方法会请求新浪微博服务器返回 JSON 对象
       msa.what=1;
    } catch (WeiboException e) {
       msa.what=0;
  }
 }
}
```

4.5.4　实现 OAuth 认证

OAuth 协议为用户资源的授权提供了一个安全、开放而又简易的标准。与以往的授权方式不同，OAuth 授权不会使第三方触及到用户的账号信息（如用户名与密码），即第三方无需使用用户的用户名与密码就可以申请获得该用户资源的授权，因此 OAuth 是安全的。

新浪微博为了实现自身的安全性，采用了 OAuth 协议认证方式。虽然下面的一段代码比较简单，但是实现了新浪微博的 OAuth 认证。

```
System.setProperty("weibo4j.oauth.consumerKey", Weibo.CONSUMER_KEY);
System.setProperty("weibo4j.oauth.consumerSecret", Weibo.CONSUMER_SECRET);
      Weibo weibo = new Weibo();
      // set callback url, desktop app please set to null
      // http://callback_url?oauth_token=xxx&oauth_verifier=xxx
      //根据 app key 第三方应用向新浪获取 requestToken
      RequestToken requestToken = weibo.getOAuthRequestToken();
      System.out.println("1.......Got request token 成功");
      System.out.println("Request token: "+ requestToken.getToken());
      System.out.println("Request token secret: "+ requestToken.getTokenSecret());
      AccessToken accessToken = null;
    //用户从新浪获取 verifier_code 如果是 Android 或 iPhone 应用可以用 callback =json&userId=xxs&pass
word=XXX
      System.out.println("Open the following URL and grant access to your account:");
      System.out.println(requestToken.getAuthorizationURL());
      BareBonesBrowserLaunch.openURL(requestToken.getAuthorizationURL());
    //用户输入验证码授权信任第三方应用
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      while (null == accessToken) {
       System.out.print("Hit enter when it's done.[Enter]:");
       String pin = br.readLine();
       System.out.println("pin: " + br.toString());
       try{
         //通过传递 requestToken 和用户验证码获取 AccessToken
             accessToken = requestToken.getAccessToken(pin);
       } catch (WeiboException te) {
             if(401 == te.getStatusCode()){
              System.out.println("Unable to get the access token.");
             }else{
              te.printStackTrace();
             }
      }
      }
      System.out.println("Got access token.");
      System.out.println("Access token: "+ accessToken.getToken());
      System.out.println("Access token secret: "+ accessToken.getTokenSecret());
    //使用 AccessToken 来操作用户的所有接口
    /* Weibo weibo=new Weibo();
      以后就可以用下面的 accessToken 访问用户的资料了
      * weibo.setToken(accessToken.getToken(), accessToken.getTokenSecret());
      //发布微博
```

```
Status status = weibo.updateStatus("test message6 ");
System.out.println("Successfully updated the status to ["
+ status.getText() + "].");
try {
            Thread.sleep(3000);
   } catch (InterruptedException e) {
// TODO Auto-generated catch block
            e.printStackTrace();
}*/
System.exit(0);
    } catch (WeiboException te) {
         System.out.println("Failed to get timeline: " + te.getMessage());
         System.exit( -1);
    } catch (IOException ioe) {
         System.out.println("Failed to read the system input.");
         System.exit( -1);
}
```

第 5 章　网络 RSS 阅读器

RSS（Really Simple Syndication）也叫聚合 RSS，是在线共享内容的一种简易方式。通常在时效性比较强的内容上使用 RSS 订阅能更快速获取信息。网站提供 RSS 输出，有利于让用户获取网站内容的最新更新。本书前面的内容中已讲解过一个简单 RSS 系统的实现流程，本章将通过一个综合实例的实现过程，详细讲解在 Android 手机平台开发一个 RSS 阅读器的方法。

5.1　实 现 流 程

知识点讲解：光盘:视频\视频讲解\第 5 章\实现流程.avi

本项目实例的功能是，在手机中显示指定 RSS 的信息，即设置显示网易博客 http://woshiyigebing-12345.blog.163.com 用户的日志信息。

本项目的具体实现流程如图 5-1 所示。

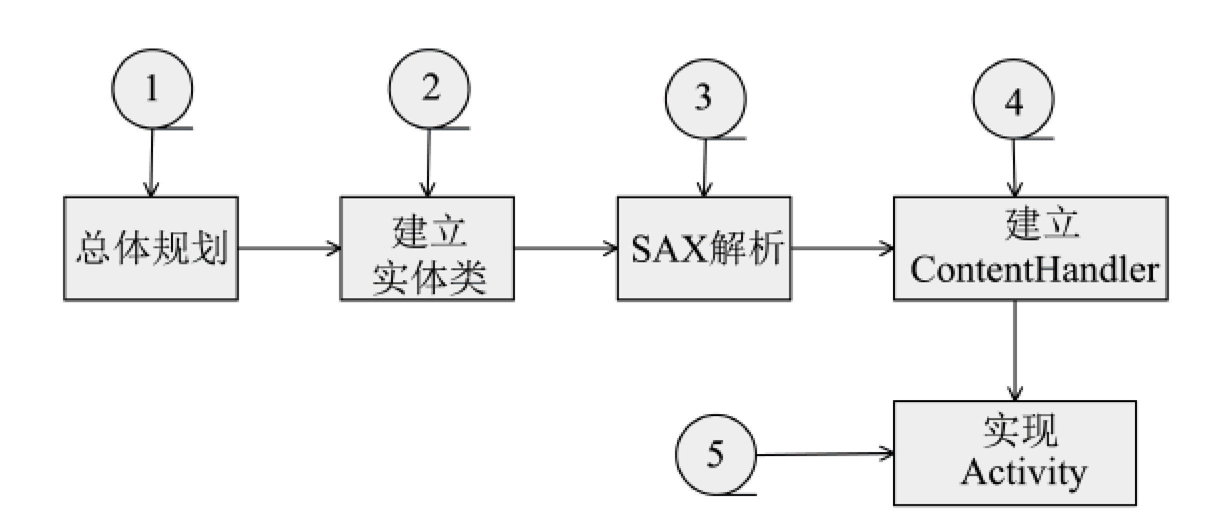


图 5-1　实现流程图

5.2　具 体 实 现

知识点讲解：光盘:视频\视频讲解\第 5 章\具体实现.avi

从本节内容开始，将着重介绍本项目的具体实现过程。详细讲解各个代码的具体实现过程，并讲解其中的技巧和要点，使读者的水平更上一层楼。

5.2.1　建立实体类

一个 RSS 文件可以被认为是由一个 RSS 的一些描述性信息和里面的 item 元素组成的，例如关于 RSS 的描述性信息。

- ☑ title：标题信息。
- ☑ link：链接信息。

☑ description：描述信息。

item 中的信息如下。

☑ title：标题信息。

☑ link：链接信息。

☑ description：描述信息。

☑ pubDate：发布的日期。

在本项目实例中需要建立如下两个实体类。

☑ RSSFeed：用于和一个 RSS 中完整的 XML 文件相对应。

☑ RSSItem：用于和一个 RSS 中 Item 标签相对应。

在解析 RSS 文件时，可以将文件里的信息解析出来放到实体类里面，这样就可以直接操作该实体类了。下面开始讲解上述两个实体类的具体实现过程。

1. RSSFeed 类

RSSFeed 类的功能是建立和一个完整 XML 文件的对应，其中，方法 addItem()用于将一个 RSSItem 添加到 RSSFeed 类中；方法 getAllItemsForListView()负责从 RSSFeed 类中生成 ListView 列表所需要的数据。RSSFeed 类的具体实现代码如下所示。

```
package com.rss_reader.data;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Vector;

public class RSSFeed
{
	private String title = null;
	private String pubdate = null;
	private int itemcount = 0;
	private List<RSSItem> itemlist;


	  public RSSFeed()
	{
		itemlist = new Vector(0);
	}
	public int addItem(RSSItem item)
	{
		itemlist.add(item);
		itemcount++;
		return itemcount;
	}
	public RSSItem getItem(int location)
	{
		return itemlist.get(location);
	}
	public List getAllItems()
	{
```

```
        return itemlist;
    }
    public List getAllItemsForListView(){
        List<Map<String, Object>> data = new ArrayList<Map<String, Object>>();
        int size = itemlist.size();
        for(int i=0;i<size;i++){
            HashMap<String, Object> item = new HashMap<String, Object>();
            item.put(RSSItem.TITLE, itemlist.get(i).getTitle());
            item.put(RSSItem.PUBDATE, itemlist.get(i).getPubDate());
            data.add(item);
        }
        return data;
    }
    int getItemCount()
    {
        return itemcount;
    }
    public void setTitle(String title)
    {
        this.title = title;
    }
    public void setPubDate(String pubdate)
    {
        this.pubdate = pubdate;
    }
    public String getTitle()
    {
        return title;
    }
    public String getPubDate()
    {
        return pubdate;
    }
}
```

2. RSSItem 类

RSSItem 类用于和一个 RSS 中 Item 标签相对应，其中的属性和 Item 中的属性一样。RSSItem 类的具体实现代码如下所示。

```
package com.rss_reader.data;

public class RSSItem
{
    public static final String TITLE="title";
    public static final String PUBDATE="pubdate";
    private String title = null;
    private String description = null;
    private String link = null;
    private String category = null;
    private String pubdate = null;
```

```
public RSSItem()
{
}
public void setTitle(String title)
{
    this.title = title;
}
Public void setDescription(String description)
{
    this.description = description;
}
Public void setLink(String link)
{
    this.link = link;
}
Public void setCategory(String category)
{
    this.category = category;
}
Public void setPubDate(String pubdate)
{
    this.pubdate = pubdate;
}
public String getTitle()
{
    return title;
}
Public String getDescription()
{
    return description;
}
Public String getLink()
{
    return link;
}
public String getCategory()
{
    return category;
}
public String getPubDate()
{
    return pubdate;
}
public String toString()
{
    if (title.length() > 20)
    {
        return title.substring(0, 42) + "...";
    }
    return title;
```

```
        }
}
```

5.2.2　主程序文件 ActivityMain.java

主程序文件 ActivityMain.java 是本项目的入口，在此 Activity 中得到了服务器端的 RSSFeed，经过解析后将里面的内容以 ListView 的形式显示出来。下面开始讲解其具体实现流程。

（1）先引入相关 class 类，然后设置目标 RSS 的源地址为 http://feed.feedsky.com/woshiyigebing12345，最后通过 showListView()方法将获取的 RSS 信息以列表形式显示出来。部分代码如下所示。

```
package com.rss_reader;

import java.net.URL;

import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.InputSource;
import org.xml.sax.XMLReader;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.AdapterView.OnItemClickListener;

import com.rss_reader.data.RSSFeed;
import com.rss_reader.data.RSSItem;
import com.rss_reader.sax.RSSHandler;

public class ActivityMain extends Activity implements OnItemClickListener
{
//      public final String RSS_URL = "http://rubyjin.cn/blog/rss";

                public final String RSS_URL = " http://feed.feedsky.com/woshiyigebing12345";

        public final String tag = "RSSReader";
        private RSSFeed feed = null;

        /** Called when the activity is first created*/

       public void onCreate(Bundle icicle) {
            super.onCreate(icicle);
            setContentView(R.layout.main);
            feed = getFeed(RSS_URL);
            showListView();

       }
```

（2）定义方法 getFeed(String urlString)，用于得到一个 RSSFeed，即从服务器端请求 RSS feed，并进行解析，将解析后的内容都放在 RSSFeed 的一个实例中。上述解析过程是通过 SAX 实现的，具体流程如下所示。

- ☑ 第 1 步：新建工厂类 SAXParserFactory。
- ☑ 第 2 步：工厂类产出一个 SAX 解析类 SAXParser。
- ☑ 第 3 步：从 SAXParser 中得到一个 XMLReader 实例，XMLReader 是一个接口，此接口中定义了一些解析 XML 的回调函数。
- ☑ 第 4 步：把编写的 Handler 注册到 XMLReader 中去。
- ☑ 第 5 步：将一个 XML 文档或资源变成一个 Java 可以处理的 InputStream 流后，解析工作开始。

方法 getFeed(String urlString)的具体代码如下所示。

```
private RSSFeed getFeed(String urlString)
{
 try
 {
        URL url = new URL(urlString);
        /*新建工厂类 SAXParserFactory*/
               SAXParserFactory factory = SAXParserFactory.newInstance();
        /*工厂类产出一个 SAX 解析类 SAXParser */
               SAXParser parser = factory.newSAXParser();
               /*从 SAXParser 中得到一个 XMLReader 实例*/
               XMLReader xmlreader = parser.getXMLReader();
               /*把编写的 Handler 注册到 XMLReader 中  */
               RSSHandler rssHandler = new RSSHandler();
               xmlreader.setContentHandler(rssHandler);

               /*将一个 XML 文档或资源变成一个 Java 可以处理的 InputStream 流后，解析工作开始*/
               InputSource is = new InputSource(url.openStream());

               xmlreader.parse(is);

               return rssHandler.getFeed();
 }
 catch (Exception ee)
 {

        return null;
 }
}
```

（3）定义方法 showListView()来列表显示获取的 RSS，这样，ListView 和一个 SimpleAdapter 实现了绑定。具体代码如下所示。

```
private void showListView()
{
      ListView itemlist = (ListView) findViewById(R.id.itemlist);
      if (feed == null)
      {
        setTitle("访问的 RSS 无效");
        return;
      }
      SimpleAdapter adapter = new SimpleAdapter(this, feed.getAllItemsForListView(),
```

```
                android.R.layout.simple_list_item_2, new String[ ] { RSSItem.TITLE,RSSItem.PUBDATE },
                new int[ ] { android.R.id.text1 , android.R.id.text2});
        itemlist.setAdapter(adapter);
        itemlist.setOnItemClickListener(this);
        itemlist.setSelection(0);

    }
```

（4）定义方法 onItemClick()，用于处理列表的单击事件，当单击后会显示此 RSS 信息的链接地址，用户单击后可以通过浏览器来到目标地址。具体代码如下所示。

```
public void onItemClick(AdapterView parent, View v, int position, long id)
{
 Intent itemintent = new Intent(this,ActivityShowDescription.class);

 Bundle b = new Bundle();
 b.putString("title", feed.getItem(position).getTitle());
 b.putString("description", feed.getItem(position).getDescription());
 b.putString("link", feed.getItem(position).getLink());
 b.putString("pubdate", feed.getItem(position).getPubDate());

 itemintent.putExtra("android.intent.extra.rssItem", b);
     startActivityForResult(itemintent, 0);
}
```

5.2.3　实现 ContentHandler

ContentHandler 是一个特殊的 SAX 接口，位于 org.xml.sax.ContentHandler。在解析 XML 时，大多数步骤都是固定不变的，但是关于 ContentHandler 的实现却是不同的。实现 ContentHandler 是解析 XML 中最重要、最关键的步骤之一，下面将开始讲解其具体实现流程。

（1）声明 RSSHandler 类，声明继承于 DefaultHandler 的类。DefaultHandler 类是一个基类，此类中最简单地实现了一个 ContentHandler，只需重写其中的重要方法即可。具体代码如下所示。

```
package com.rss_reader.sax;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

import android.util.Log;

import com.rss_reader.data.RSSFeed;
import com.rss_reader.data.RSSItem;

public class RSSHandler extends DefaultHandler
{

      RSSFeed rssFeed;
      RSSItem rssItem;
      String lastElementName = "";
      final int RSS_TITLE = 1;
      final int RSS_LINK = 2;
```

```
        final int RSS_DESCRIPTION = 3;
        final int RSS_CATEGORY = 4;
        final int RSS_PUBDATE = 5;

        int currentstate = 0;

        public RSSHandler()
        {
        }

        public RSSFeed getFeed()
        {
                return rssFeed;
        }
```

（2）分别重写 startDocument()和 endDocument()，通常将正式解析前的初始化工作放到 startDocument()中，将一些收尾性工作放到 endDocument()中，具体代码如下所示。

```
public void startDocument() throws SAXException
{
        rssFeed = new RSSFeed();
        rssItem = new RSSItem();

}
public void endDocument() throws SAXException
{
}
```

（3）重写 startElement()，当 XML 解析器遇到 XML 文档流中的 tag 时，将会调用此函数。在此函数内部通常是通过参数 localName 判断并进行一些操作处理的，具体代码如下所示。

```
public void startElement(String namespaceURI, String localName,String qName, Attributes atts) throws
SAXException
{
        if (localName.equals("channel"))
        {
                currentstate = 0;
                return;
        }
        if (localName.equals("item"))
        {
                rssItem = new RSSItem();
                return;
        }
        if (localName.equals("title"))
        {
                currentstate = RSS_TITLE;
                return;
        }
        if (localName.equals("description"))
        {
                currentstate = RSS_DESCRIPTION;
                return;
        }
```

```
        if (localName.equals("link"))
        {
            currentstate = RSS_LINK;
            return;
        }
        if (localName.equals("category"))
        {
            currentstate = RSS_CATEGORY;
            return;
        }
        if (localName.equals("pubDate"))
        {
            currentstate = RSS_PUBDATE;
            return;
        }

        currentstate = 0;
}
```

（4）重写 endElement()，此方法和 startElement()方法相对应，当解析 tag 完毕后执行此方法。如果解析一个 item 节点结束，就将 RSSItem 添加到 RSSFeed 中去，具体代码如下所示。

```
public void endElement(String namespaceURI, String localName, String qName) throws SAXException
{

        //如果解析一个 item 节点结束，就将 rssItem 添加到 RSSFeed 中
        if (localName.equals("item"))
        {
            rssFeed.addItem(rssItem);
            return;

        }
}
```

（5）重写 characters()，此方法是一个回调方法，当解析完 startElement()方法后，解析完节点内容后会执行此方法，并且参数 ch[ ]就是节点的内容，具体代码如下所示。

```
public void characters(char ch[ ], int start, int length)
{
        String theString = new String(ch,start,length);

        switch (currentstate)
        {
            case RSS_TITLE:
                rssItem.setTitle(theString);
                currentstate = 0;
                break;
            case RSS_LINK:
                rssItem.setLink(theString);
                currentstate = 0;
                break;
            case RSS_DESCRIPTION:
                rssItem.setDescription(theString);
                currentstate = 0;
```

```
                break;
            case RSS_CATEGORY:
                rssItem.setCategory(theString);
                currentstate = 0;
                break;
            case RSS_PUBDATE:
                rssItem.setPubDate(theString);
                currentstate = 0;
                break;
            default:
                return;
        }
}
```

5.2.4 主程序文件 ActivityShowDescription.java

主程序文件 ActivityShowDescription.java 的功能是显示某列表信息的详细信息。当单击列表中的某一项后，会进入到此界面。如果程序出错，则 content 显示出错提示；运行正确则在 content 中分别显示 title、pubdate 和 description。具体代码如下所示。

```
package com.rss_reader;

import android.app.Activity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import android.content.Intent;
import android.view.*;

public class ActivityShowDescription extends Activity {
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.showdescription);
        String content = null;
        Intent startingIntent = getIntent();

        if (startingIntent != null) {
            Bundle bundle = startingIntent
                    .getBundleExtra("android.intent.extra.rssItem");
            if (bundle == null) {
                content = "不好意思程序出错啦";
            } else {
                content = bundle.getString("title") + "\n\n"
                        + bundle.getString("pubdate") + "\n\n"
                        + bundle.getString("description").replace('\n', ' ')
                        + "\n\n 详细信息请访问以下网址:\n" + bundle.getString("link");
            }
        } else {
            content = "不好意思程序出错啦";

        }
```

```
        TextView textView = (TextView) findViewById(R.id.content);
        textView.setText(content);

        Button backbutton = (Button) findViewById(R.id.back);

        backbutton.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                finish();
            }
        });
    }
}
```

5.2.5　主布局文件 main.xml

主布局文件 main.xml 用于定义系统初始主界面，即列表显示获取的 RSS 信息，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<ListView
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:id="@+id/itemlist"

     />
</LinearLayout>
```

5.2.6　详情主布局文件 showdescription.xml

当用户单击列表信息后，会进入信息详情界面，此界面是由布局文件 showdescription.xml 定义的，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:autoLink="all"
    android:text=""
    android:id="@+id/content"
    android:layout_weight="1.0"
    />
```

```
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="返回"
    android:id="@+id/back"
    />

</LinearLayout>
```

至此，整个实例介绍完毕。运行后将获取指定 RSS 中的信息，如图 5-2 所示。单击某条信息后会显示此信息的相关描述性信息，如图 5-3 所示。



图 5-2　初始效果



图 5-3　详情界面

单击图 5-3 中间的超链接后，能够显示此条 RSS 的详细信息，如图 5-4 所示。

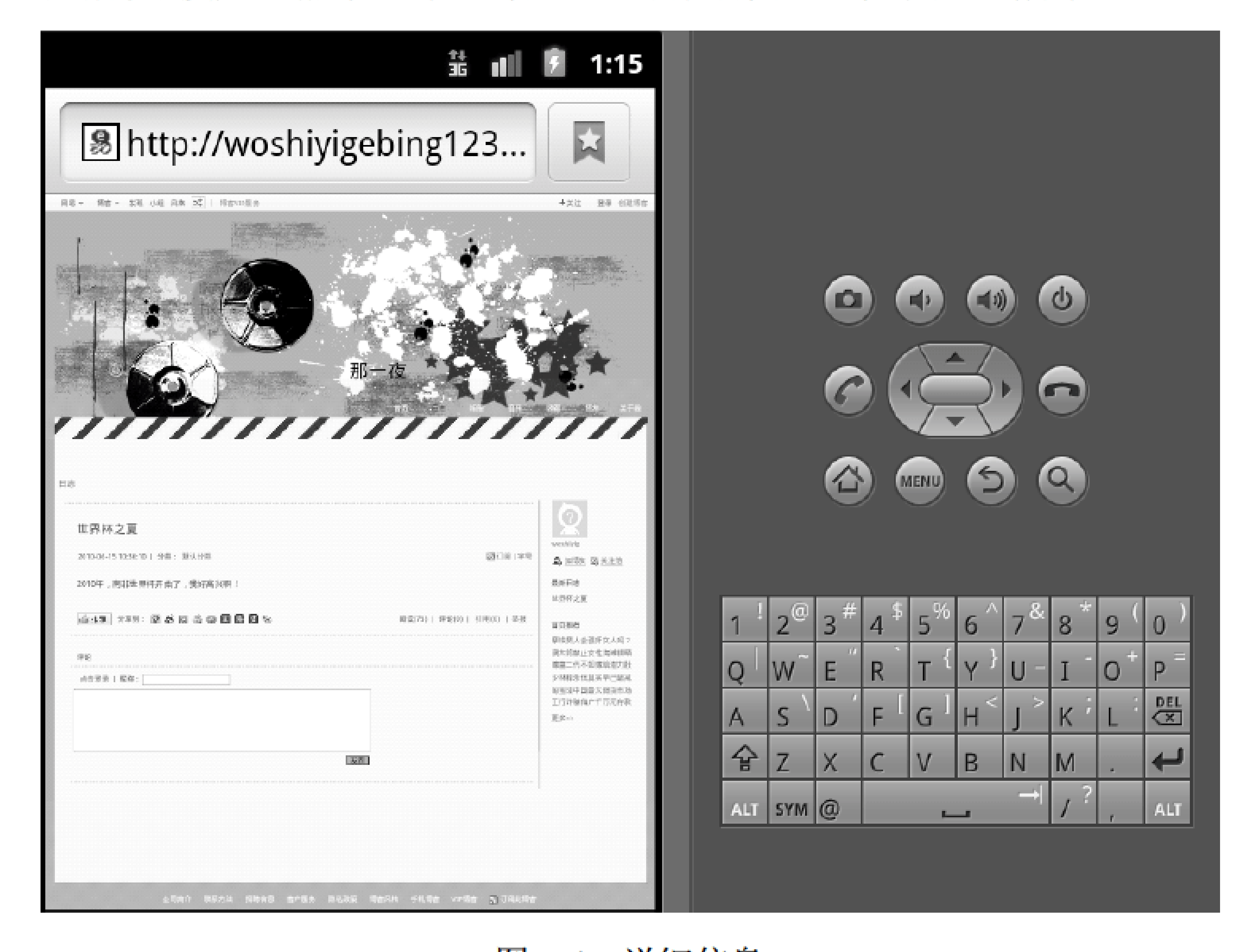


图 5-4　详细信息

本实例默认显示的是博客 http://woshiyigebing12345.blog.163.com/中的信息。读者也可以指定显示其他 RSS 信息，在使用时可以登录 http://www.feedsky.com/来设置不同的 RSS 订阅。具体设置流程如下。

（1）打开 http://www.feedsky.com/主界面，如图 5-5 所示。

（2）在图 5-5 顶部的文本框中输入要显示信息的博客地址、Feed 地址或 QQ 号码，然后单击“下一步”按钮，如图 5-6 所示。

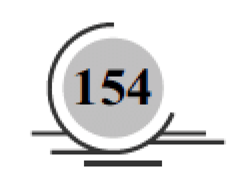




图 5-5　feedsky.com 主界面　　　　图 5-6　输入设置的博客、QQ 或 Feed 地址

（3）在弹出界面中分别输入 Feed 名称、Feed 描述和 Tag，并设定永久性 Feed 地址，如图 5-7 所示。



图 5-7　添加 Feed 界面

图 5-7 中的永久 Feed 地址就是 RSS 的源地址，这样就可以将此地址添加到实例中，从而显示此地址的 RSS 资源信息，也就是显示博客 http://woshiyigebing12345.blog.163.com/中的信息。

5.3　打包、签名和发布

知识点讲解：光盘:视频\视频讲解\第 5 章\打包、签名和发布.avi

当一个 Android 项目开发完毕后，需要打包和签名处理，这样才能放到手机中使用，当然也可以发布到 Market 上去赚钱。下面开始讲解打包、签名、发布 Android 程序的具体过程。

5.3.1　申请会员

去 Market 申请成为会员，具体流程如下。

（1）登录 http://market.android/publish/signup，如图 5-8 所示。



图 5-8 登录 Market

（2）单击 Create an account now 超链接，来到注册页面，如图 5-9 所示。

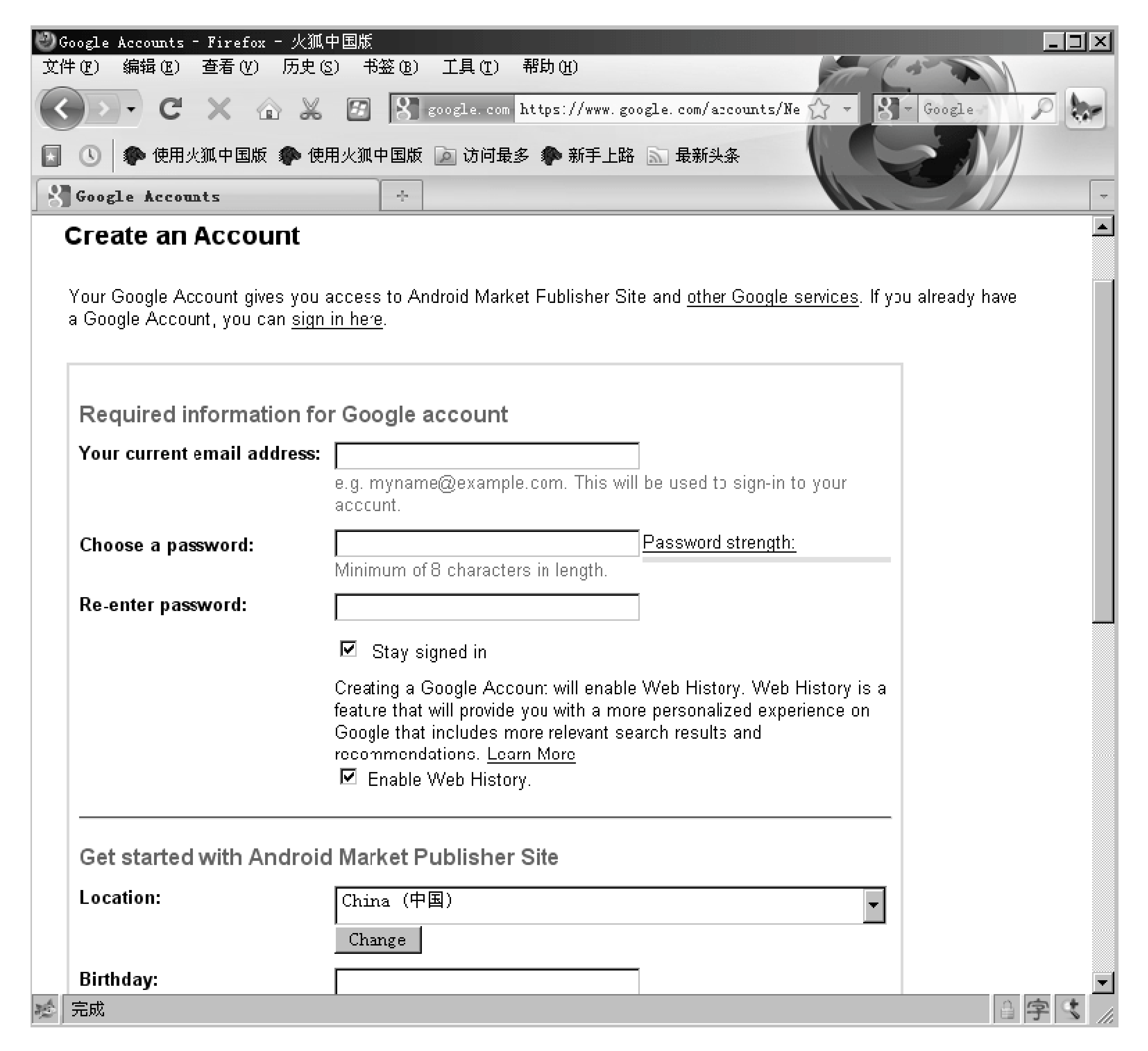
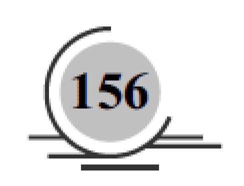


图 5-9 注册界面

（3）单击同意协议后来到下一步页面，在此输入手机号码，如图 5-10 所示。



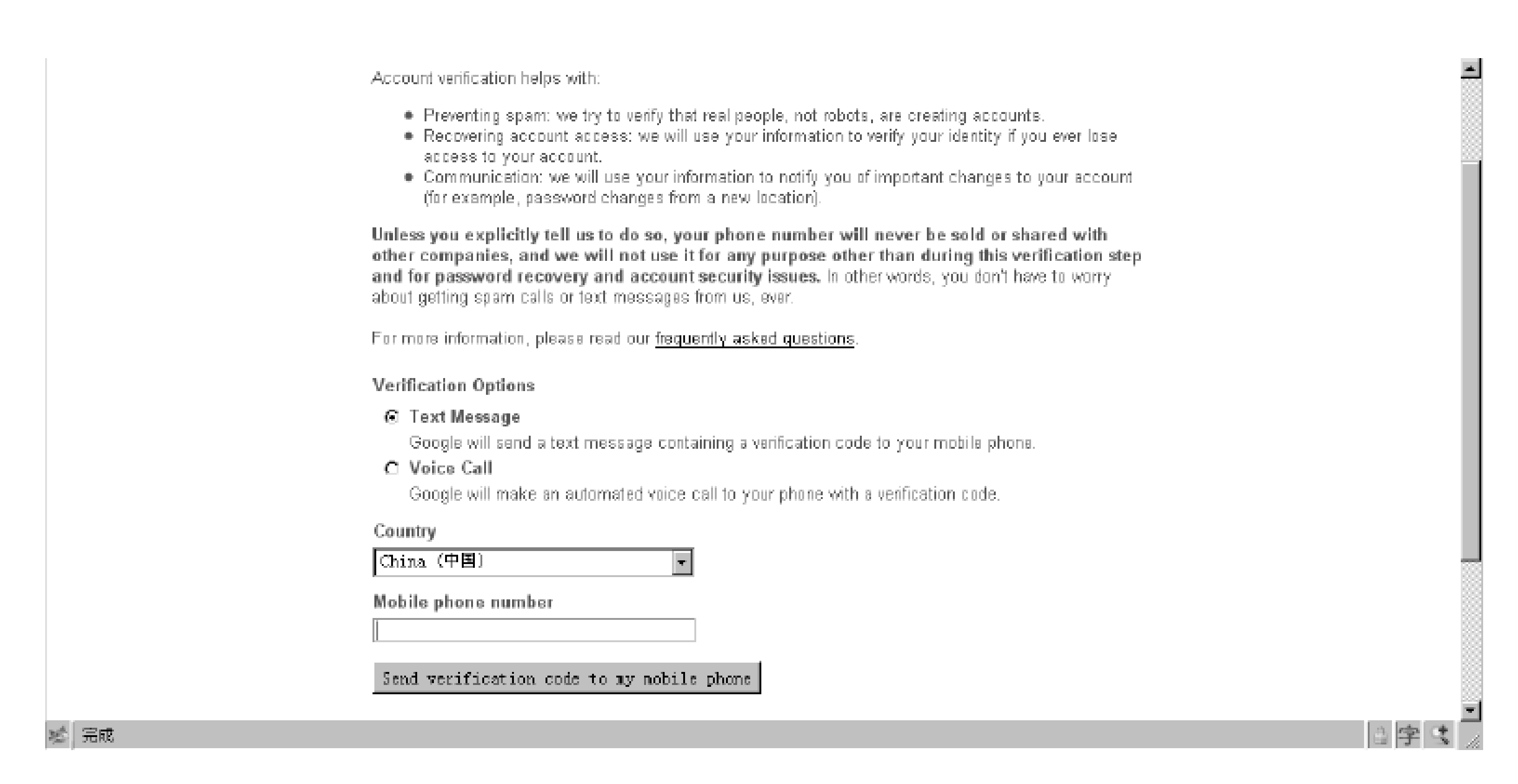


图 5-10　输入手机号码

（4）在新页面中输入手机获取的验证码，如图 5-11 所示。



图 5-11　输入验证码

（5）验证通过后，在新页面中继续输入信息，如图 5-12 所示。

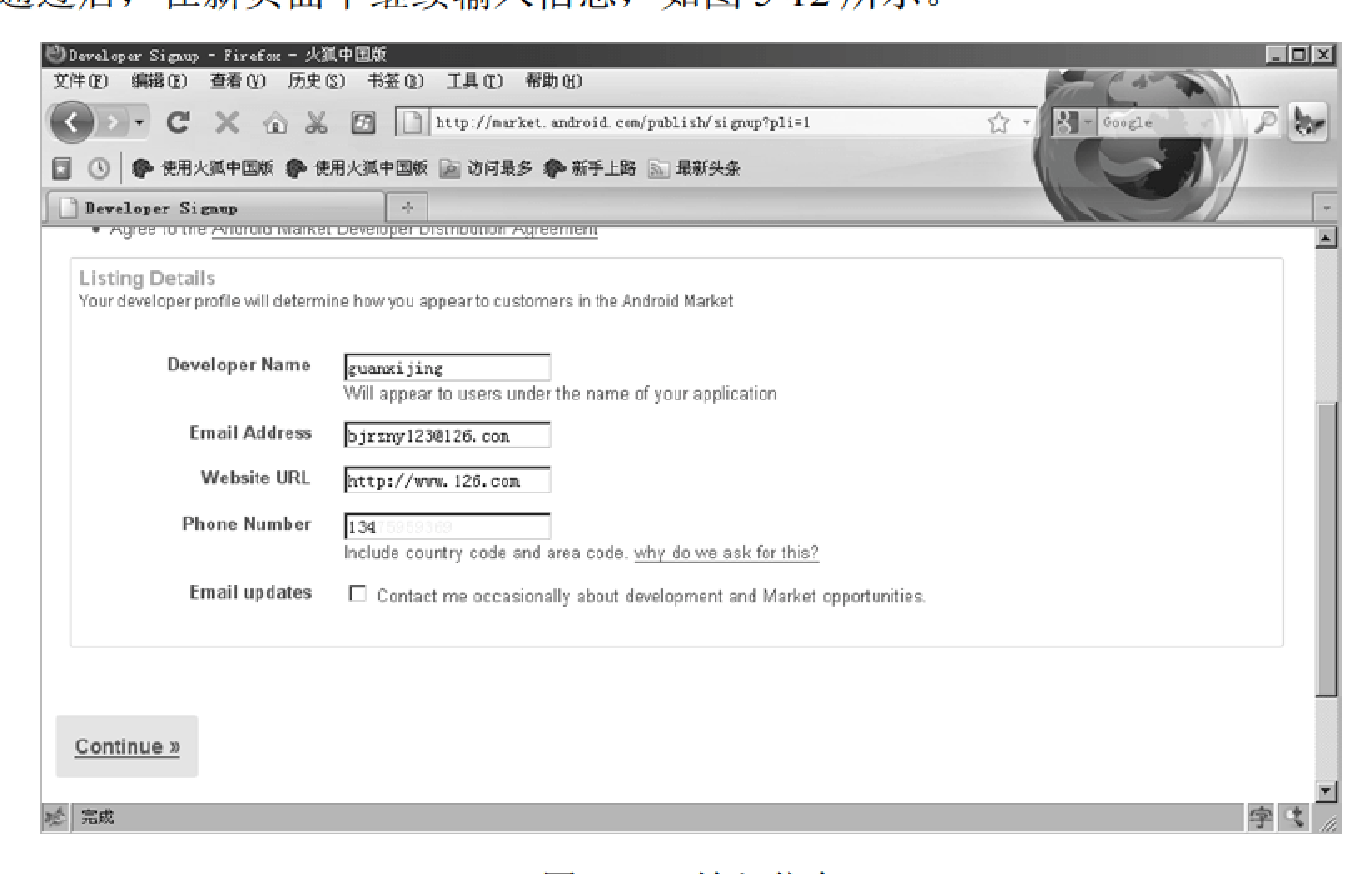


图 5-12　输入信息

（6）单击 Continue 按钮后，提示需要花费 25 美元，支付后才能成为正式会员，如图 5-13 所示。



图 5-13　需要支付界面

（7）单击 Google Checkout 按钮来到支付界面，如图 5-14 所示。

图 5-14　支付界面

在此输入信用卡信息，完成支付后即可成为正式会员。

5.3.2　生成签名文件

Android 程序的签名和 Symbian 类似都可以自签名（Self-signed），但是在 Android 平台中证书初期还显得形同虚设，平时开发时通过 ADB 接口上传的程序会自动被签有 Debug 权限的程序。

Android 签名文件的制作方法有两种。

第一种：命令行生成。具体流程如下。

（1）CMD 命令如下。

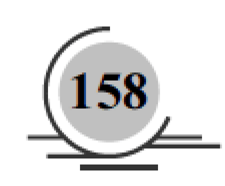
```
keytool -genkey -alias android123.keystore -keyalg RSA -validity 20000 -keystore android123.keystore
```

然后依次提示用户输入如下信息。

输入 keystore 密码：[密码不回显]

再次输入新密码：[密码不回显]

您的名字与姓氏是什么？



[Unknown]：　android123
您的组织单位名称是什么？
[Unknown]：　www.android123.com.cn
您的组织名称是什么？
[Unknown]：　www.android123.com.cn
您所在的城市或区域名称是什么？
[Unknown]：　New York
您所在的州或省份名称是什么？
[Unknown]：　New York
该单位的两字母国家代码是什么？
[Unknown]：　CN
CN=android123, OU=www.android123.com.cn, O=www.android123.com.cn, L=New York, ST =New York, C=CN　正确吗？
[否]：Y
输入<android123.keystore>的主密码，如果和 keystore 密码相同，按 Enter 键。

其中，参数-validity 为证书有效天数，这里设为 200 天。还有在输入密码时没有回显，只输入就可以了，一般位数建议使用 20 位，最后需要记下来后面还要用。接下来就可以为 APK 文件签名了。

（2）执行以下代码。

```
jarsigner -verbose -keystore android123.keystore -signedjar android123_signed.apk android123.apk android123. keystore
```

这样就可以生成签名的 APK 文件，假设输入文件 android123.apk，则最终生成 android123_signed.apk，为 Android 签名后的 APK 执行文件。

keytool 用法和 jarsigner 用法总结如下。

（1）keytool 用法

```
-certreq [-v] [-protected]
          [-alias <别名>] [-sigalg <sigalg>]
          [-file <csr_file>] [-keypass <密钥库口令>]
          [-keystore <密钥库>] [-storepass <存储库口令>]
          [-storetype <存储类型>] [-providername <名称>]
          [-providerclass <提供方类名称> [-providerarg <参数>]] ...
          [-providerpath <路径列表>]

-changealias [-v] [-protected] -alias <别名> -destalias <目标别名>
          [-keypass <密钥库口令>]
          [-keystore <密钥库>] [-storepass <存储库口令>]
          [-storetype <存储类型>] [-providername <名称>]
          [-providerclass <提供方类名称> [-providerarg <参数>]] ...
          [-providerpath <路径列表>]

-delete [-v] [-protected] -alias <别名>
          [-keystore <密钥库>] [-storepass <存储库口令>]
          [-storetype <存储类型>] [-providername <名称>]
          [-providerclass <提供方类名称> [-providerarg <参数>]] ...
```

```
                    [-providerpath <路径列表>]

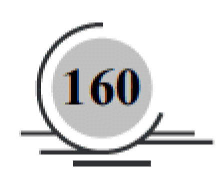
-exportcert [-v] [-rfc] [-protected]
                    [-alias <别名>] [-file <认证文件>]
                    [-keystore <密钥库>] [-storepass <存储库口令>]
                    [-storetype <存储类型>] [-providername <名称>]
                    [-providerclass <提供方类名称> [-providerarg <参数>]] ...
                    [-providerpath <路径列表>]

-genkeypair [-v] [-protected]
                    [-alias <别名>]
                    [-keyalg <keyalg>] [-keysize <密钥大小>]
                    [-sigalg <sigalg>] [-dname <dname>]
                    [-validity <valDays>] [-keypass <密钥库口令>]
                    [-keystore <密钥库>] [-storepass <存储库口令>]
                    [-storetype <存储类型>] [-providername <名称>]
                    [-providerclass <提供方类名称> [-providerarg <参数>]] ...
                    [-providerpath <路径列表>]

-genseckey [-v] [-protected]
                    [-alias <别名>] [-keypass <密钥库口令>]
                    [-keyalg <keyalg>] [-keysize <密钥大小>]
                    [-keystore <密钥库>] [-storepass <存储库口令>]
                    [-storetype <存储类型>] [-providername <名称>]
                    [-providerclass <提供方类名称> [-providerarg <参数>]] ...
                    [-providerpath <路径列表>]

-help
-importcert [-v] [-noprompt] [-trustcacerts] [-protected]
                    [-alias <别名>]
                    [-file <认证文件>] [-keypass <密钥库口令>]
                    [-keystore <密钥库>] [-storepass <存储库口令>]
                    [-storetype <存储类型>] [-providername <名称>]
                    [-providerclass <提供方类名称> [-providerarg <参数>]] ...
                    [-providerpath <路径列表>]

-importkeystore [-v]
                    [-srckeystore <源密钥库>] [-destkeystore <目标密钥库>]
                    [-srcstoretype <源存储类型>] [-deststoretype <目标存储类型>]
                    [-srcstorepass <源存储库口令>] [-deststorepass <目标存储库口令>]
                    [-srcprotected] [-destprotected]
```



```
            [-srcprovidername <源提供方名称>]
            [-destprovidername <目标提供方名称>]
            [-srcalias <源别名> [-destalias <目标别名>]
            [-srckeypass <源密钥库口令>] [-destkeypass <目标密钥库口令>]]
            [-noprompt]
            [-providerclass <提供方类名称> [-providerarg <参数>]] ...
            [-providerpath <路径列表>]

-keypasswd [-v] [-alias <别名>]
            [-keypass <旧密钥库口令>] [-new <新密钥库口令>]
            [-keystore <密钥库>] [-storepass <存储库口令>]
            [-storetype <存储类型>] [-providername <名称>]
            [-providerclass <提供方类名称> [-providerarg <参数>]] ...
            [-providerpath <路径列表>]

-list [-v | -rfc] [-protected]
            [-alias <别名>]
            [-keystore <密钥库>] [-storepass <存储库口令>]
            [-storetype <存储类型>] [-providername <名称>]
            [-providerclass <提供方类名称> [-providerarg <参数>]] ...
            [-providerpath <路径列表>]

-printcert [-v] [-file <认证文件>]

-storepasswd [-v] [-new <新存储库口令>]
            [-keystore <密钥库>] [-storepass <存储库口令>]
            [-storetype <存储类型>] [-providername <名称>]
            [-providerclass <提供方类名称> [-providerarg <参数>]] ...
            [-providerpath <路径列表>]
```

（2）jarsigner 用法

```
[选项] jar 文件别名
jarsigner -verify [选项] jar 文件

[-keystore <url>]          密钥库位置
[-storepass <口令>]        用于密钥库完整性的口令
[-storetype <类型>]        密钥库类型
[-keypass <口令>]          专用密钥的口令（如果不同）
[-sigfile <文件>]          .SF/.DSA 文件的名称
[-signedjar <文件>]        已签名的 JAR 文件的名称
[-digestalg <算法>]        摘要算法的名称
[-sigalg <算法>]           签名算法的名称
```

[-verify]	验证已签名的 JAR 文件
[-verbose]	签名/验证时输出详细信息
[-certs]	输出详细信息和验证时显示证书
[-tsa <url>]	时间戳机构的位置
[-tsacert <别名>]	时间戳机构的公共密钥证书
[-altsigner <类>]	替代的签名机制的类名
[-altsignerpath <路径列表>]	替代的签名机制的位置
[-internalsf]	在签名块内包含 .SF 文件
[-sectionsonly]	不计算整个清单的散列
[-protected]	密钥库已保护验证路径
[-providerName <名称>]	提供者名称
[-providerClass <类>	加密服务提供者的名称
[-providerArg <参数>]] ...	主类文件和构造函数参数

第二种：Eclipse 的 ADT 生成。

实际上，使用 Eclipse 可以更加直观、方便地生成签名文件，具体流程如下。

（1）右击 Eclipse 项目名，依次选择 Android Tools | Export Signed Application Package…命令，如图 5-15 所示。

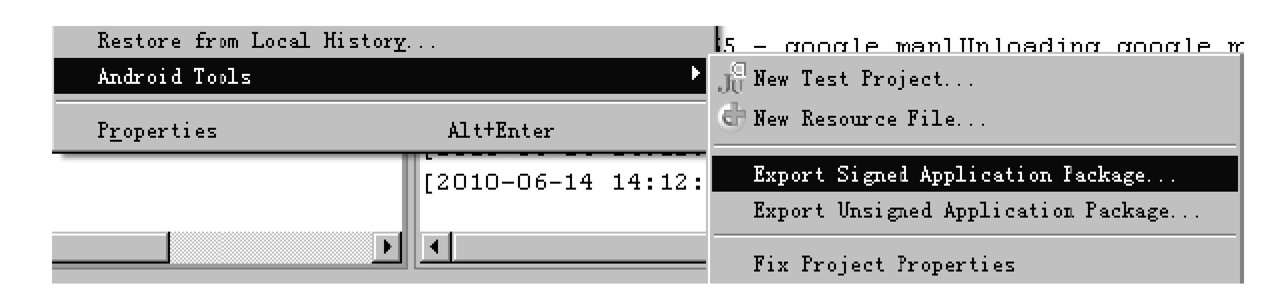


图 5-15 选择导出

（2）在弹出界面中选择要导出的项目，如图 5-16 所示。

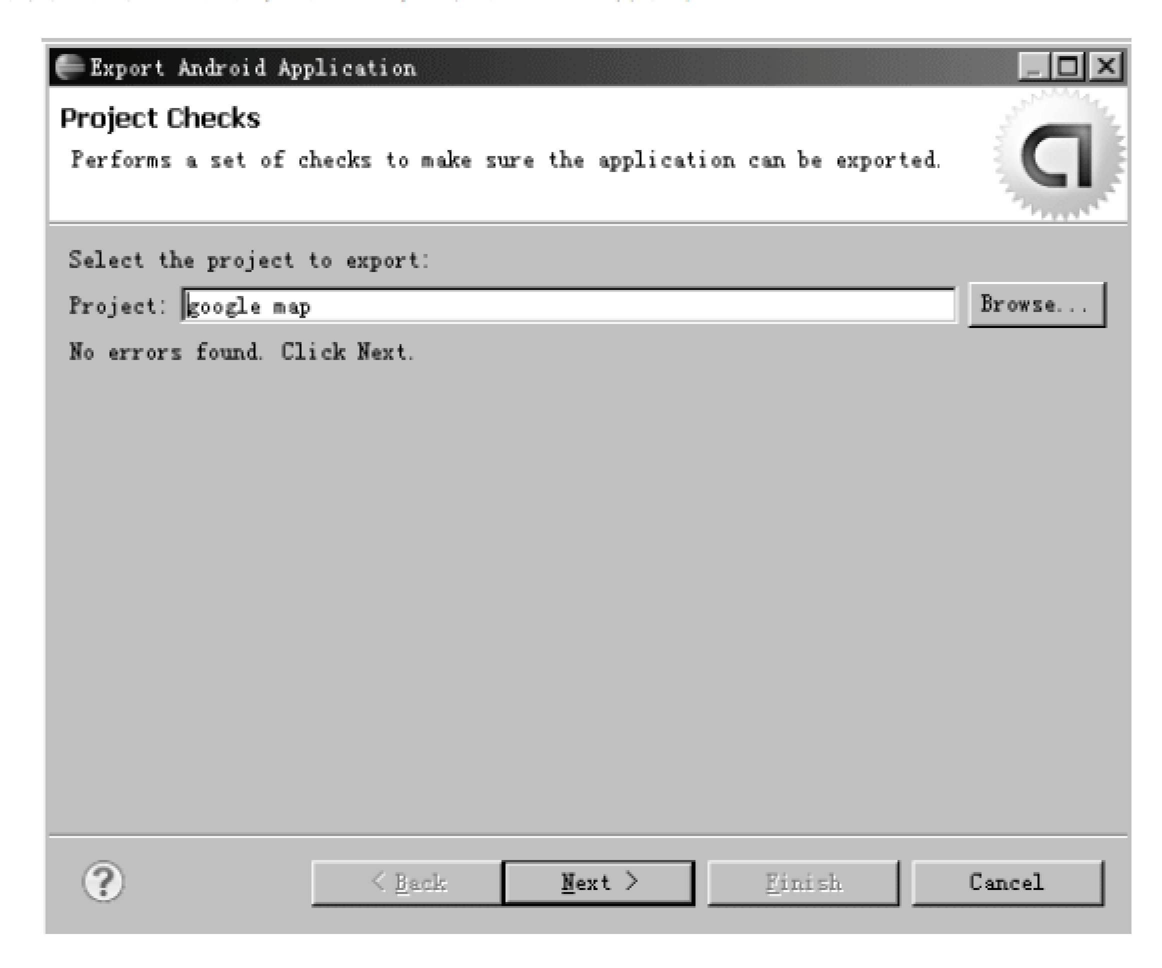
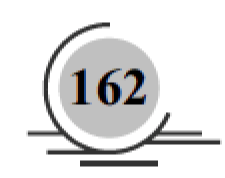


图 5-16 选择要导出的项目

（3）单击 Next 按钮，在弹出的界面中选中 Create new keystore 单选按钮，然后分别输入文件名和密码，如图 5-17 所示。

（4）单击 Next 按钮，在弹出的界面中输入签名文件路径，如图 5-18 所示。



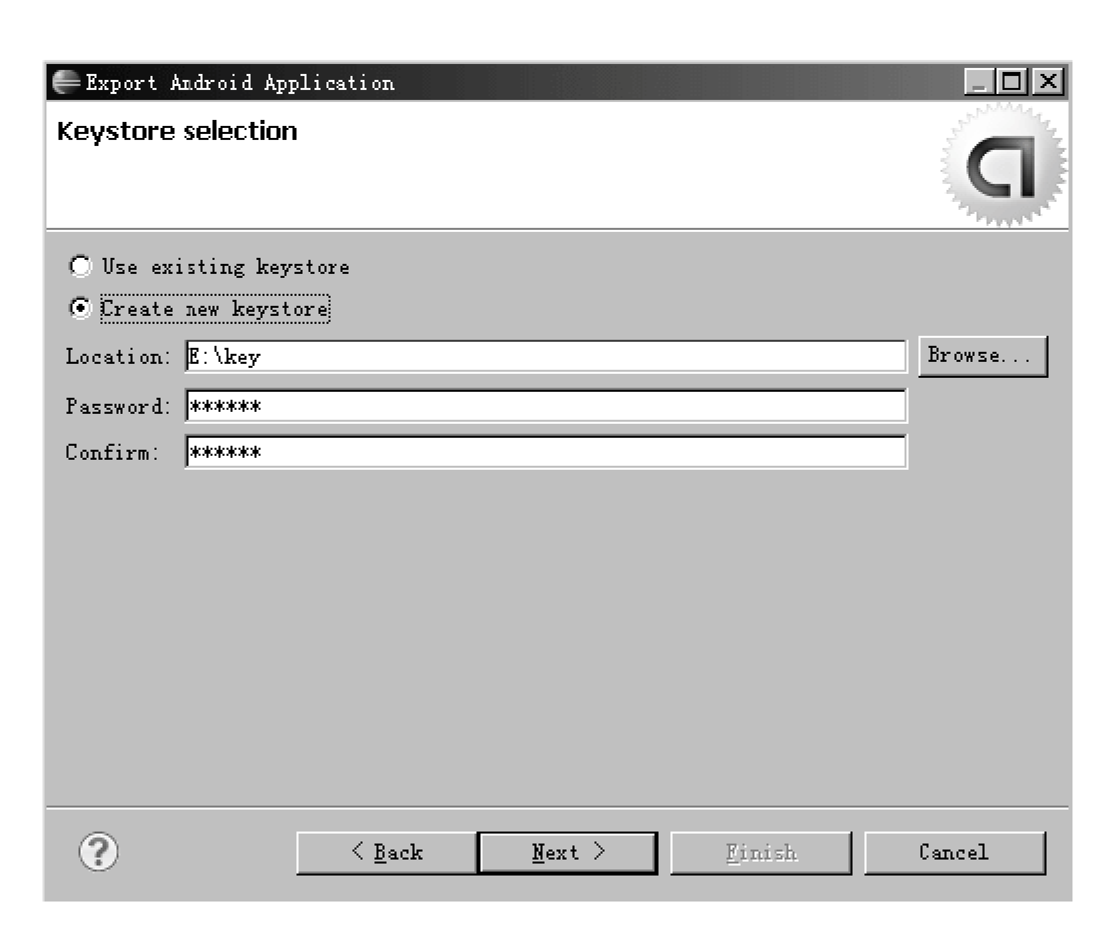


图 5-17　文件名和密码

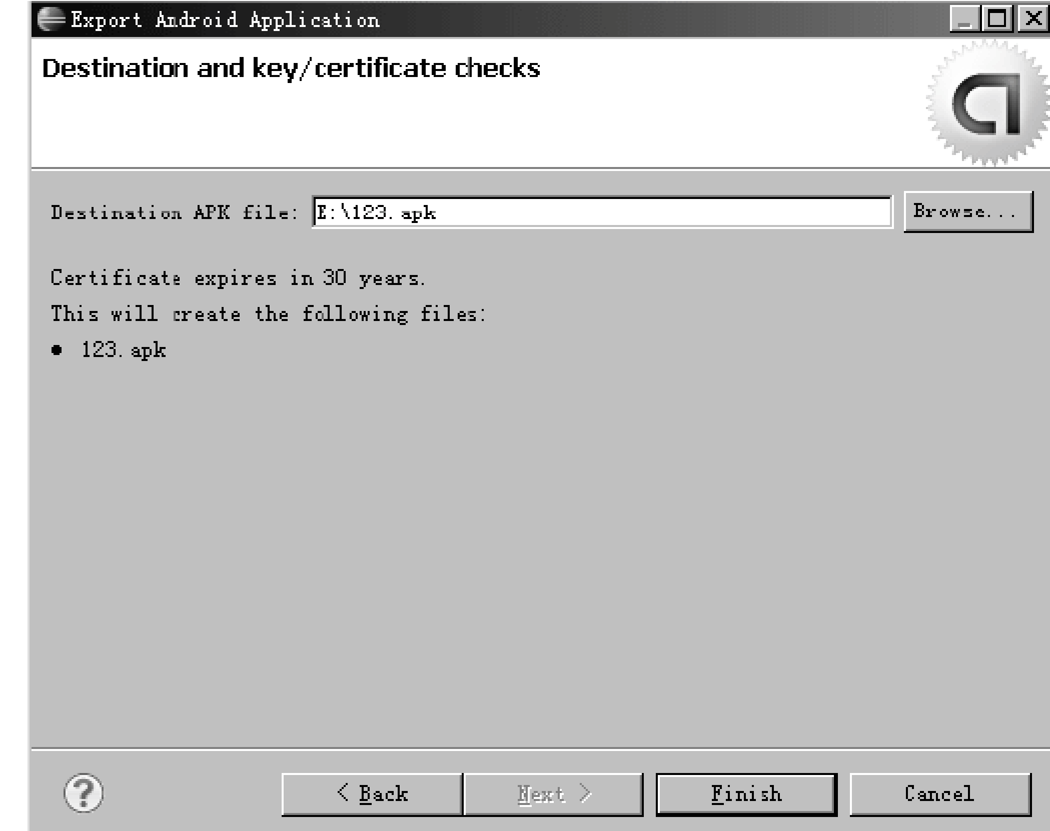


图 5-18　输入信息

（5）单击 Next 按钮，在弹出的界面中依次输入签名文件的相关信息，如图 5-19 所示。

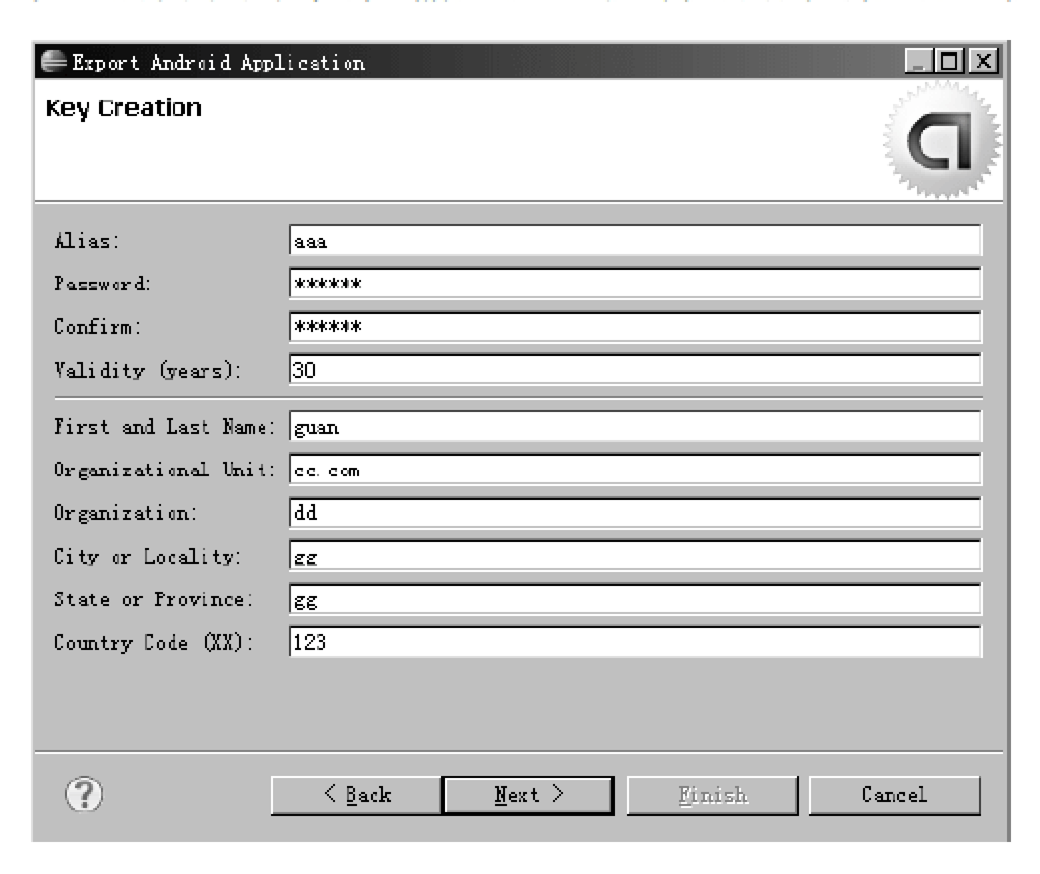


图 5-19　输入信息

（6）单击 Next 按钮后完成签名文件的创建。

5.3.3　使用签名文件

生成签名文件后，就可以使用，在此也有两种方式。

第一种：命令行。

（1）假设生成的签名文件是 ChangeBackgroundWidget.apk，则最终生成 ChangeBackgroundWidget_signed.apk 为 Android 签名后的 APK 执行文件。

输入以下命令。

```
jarsigner -verbose -keystore ChangeBackgroundWidget.keystore -signedjar ChangeBackgroundWidget_signed.
apk ChangeBackgroundWidget.apk ChangeBackgroundWidget.keystore
```

上面命令中间不换行。

（2）按 Enter 键，根据提示输入密钥库的口令短语（即密码），详细信息如下。

```
正在添加：META-INF/MANIFEST.MF
正在添加：META-INF/CHANGEBA.SF
```

```
正在添加：META-INF/CHANGEBA.RSA
正在签名：res/drawable/icon.png
正在签名：res/drawable/icon_audio.png
正在签名：res/drawable/icon_exit.png
正在签名：res/drawable/icon_folder.png
正在签名：res/drawable/icon_home.png
正在签名：res/drawable/icon_img.png
正在签名：res/drawable/icon_left.png
正在签名：res/drawable/icon_mantou.png
正在签名：res/drawable/icon_other.png
正在签名：res/drawable/icon_pause.png
正在签名：res/drawable/icon_play.png
正在签名：res/drawable/icon_return.png
正在签名：res/drawable/icon_right.png
正在签名：res/drawable/icon_set.png
正在签名：res/drawable/icon_text.png
正在签名：res/drawable/icon_xin.png
正在签名：res/layout/fileitem.xml
正在签名：res/layout/filelist.xml
正在签名：res/layout/main.xml
正在签名：res/layout/widget.xml
正在签名：res/xml/widget_info.xml
正在签名：AndroidManifest.xml
正在签名：resources.arsc
正在签名：classes.dex
```

通过上述过程处理后，即可将未签名文件 ChangeBackgroundWidget.apk 重命名为 ChangeBackground-Widget_signed.apk。

在上述方式中，读者可能会遇到以下问题。

☑ 问题一：jarsigner 无法打开 JAR 文件 ChangeBackgroundWidget.apk。

解决方法：将要进行签名的 APK 放到对应的文件下，把要签名的 ChangeBackgroundWidget.apk 放到 JDK 的 bin 文件里。

☑ 问题二：jarsigner 无法对 JAR 进行签名：java.util.zip.ZipException: invalid entry compressed size (expected 1598 but got 1622 bytes)。

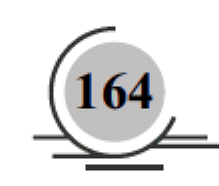
解决方法一：Android 开发网提示这些问题主要是由于资源文件造成的，对于 Android 开发来说应该检查 res 文件夹中的文件，逐个排查。这个问题可以通过升级系统的 JDK 和 JRE 版本来解决。

解决方法二：这是因为默认给 APK 做了 debug 签名，所以无法做新的签名，这时必须在工程名上右击，并选择 Android Tools | Export Unsigned Application Package 命令，或者从 AndroidManifest.xml 的 Exporting 上操作，效果相同。

然后再基于这个导出的 unsigned Apk 做签名，导出时最好将其目录选在之前产生 keystore 的目录下，这样操作起来就方便了。

5.3.4 发布

发布的过程比较简单，来到 Market，登录个人中心，上传签名后的文件即可，具体操作流程在 Market 站点上有详细介绍说明。为节省本书的篇幅，此处不再做详细介绍。



第 6 章　开发一个音乐播放器

随着社会生活节奏的加快，硬件移动设备越来越先进，人们对移动设备的要求也越来越高，从以前的追求技术到现在的追求视觉，逐步提高了对系统的要求。本章的音乐播放器实例采用了 Android 开源系统技术，利用 Java 语言和 Eclipse 编辑工具对播放器进行编写。同时给出了详细的系统设计过程、部分界面图及主要功能运行流程图，本章还对开发过程中遇到的问题和解决方法进行了详细的讨论。本章设计的音乐播放器实例集播放、暂停、停止、上一首、下一首、音量调节、歌词显示等功能于一体，性能良好，在 Android 系统中能独立运行。该播放器还拥有对手机文件浏览器的访问功能、歌曲播放模式以及歌词开闭状态的友好设置。因为本播放器只限于应用层程序的探讨，所以对具体的压缩算法不作深究。

6.1　项 目 介 绍

知识点讲解：光盘:视频\视频讲解\第 6 章\项目介绍.avi

手机市场的迅速发展，使得手机操作系统也出现了不同分类，现在的市场上主要有 3 个手机操作系统，即 Windows Mobile、Symbian 以及谷歌的 Android 操作系统，其中占有开放源代码优势的 Android 系统有最大的发展前景。那么能否在手机上拥有自己编写的个性音乐播放器呢？答案是完全可以！谷歌旗下的 Android 系统就可以做到，本章讲解的音乐播放器实例就是基于谷歌 Android 手机平台的播放器。

6.1.1　项目背景介绍

随着计算机的广泛运用，手机市场的迅速发展，各种音频视频资源也在网上广为流传，这些资源看似平常，但已经渐渐成为人们生活中不可或缺的一部分了。于是各种手机播放器也紧跟着发展起来，但是很多播放器一味追求外观花哨、功能庞大，对用户的手机造成了很多资源浪费，例如 CPU、内存等的占用率过高，在用户需要多任务操作时，受到不小的影响，带来了许多不便，而对于大多数普通用户，许多功能用不上，形同虚设。针对以上各种弊端，选择了开发多语种的音频视频播放器，将各种性能优化，继承播放器的常用功能，满足一般用户（如听歌、看电影）的需求，除了能播放常见格式的语音视频文件高级功能外，还能播放 RMVB 格式的视频文件。此外，还能支持中文、英文等语言界面。

开发一个播放器需要研究市场上流行的各种手机播放器，了解其各自的插件及编码方式，还有各种播放器播放的特别格式文件，分析各种编码的优缺点以及各种播放器本身存在的缺陷和特点，编写出功能实用、使用方便快捷的播放器。目前已经实现的功能有能播放常见音频文件的功能，例如 MP3 和 WAV 等，拥有播放菜单，能选择播放清单，具备一般播放器的功能，如快进、快退、音量调节等。播放模式也比较完善，例如，有单曲播放、顺序播放、循环播放和随机播放等模式。

6.1.2　项目的目的

由于生活节奏快，工作压力大，欣赏音乐成为舒缓压力的一种方式，本项目的目的是开发一个可以播放主流格式音乐文件的播放器，这个播放器实例的主要功能是播放 MP3、WAV 等多种格式的音乐文件，并

且能够控制播放、暂停、停止、上一曲、下一曲音乐，也具备音量调节功能，并且具有较好的视觉外观，还具有播放列表和歌曲文件的管理操作等多种播放控制功能，界面简明，操作简单。

本项目是一款基于 Android 手机平台的音乐播放器，使 Android 手机拥有个性的多媒体播放器，使手机显得更生动灵活化，让手机主人随时随地处于音乐视频的旋律之中，生活更加多样化，也使设计者更加熟悉 Android 的技术。

6.2 系统需求分析

知识点讲解：光盘:视频\视频讲解\第 6 章\系统需求分析.avi

根据项目的目标，可获得项目系统的基本需求，下面从不同角度来描述系统的需求，并且使用用例图来描述，系统的功能需求分成 4 部分来概括，即播放器的基本控制需要、播放列表管理需求、播放器友好性需求和播放器扩展卡需求。

6.2.1 构成模块

本项目系统的构成模块如图 6-1 所示。

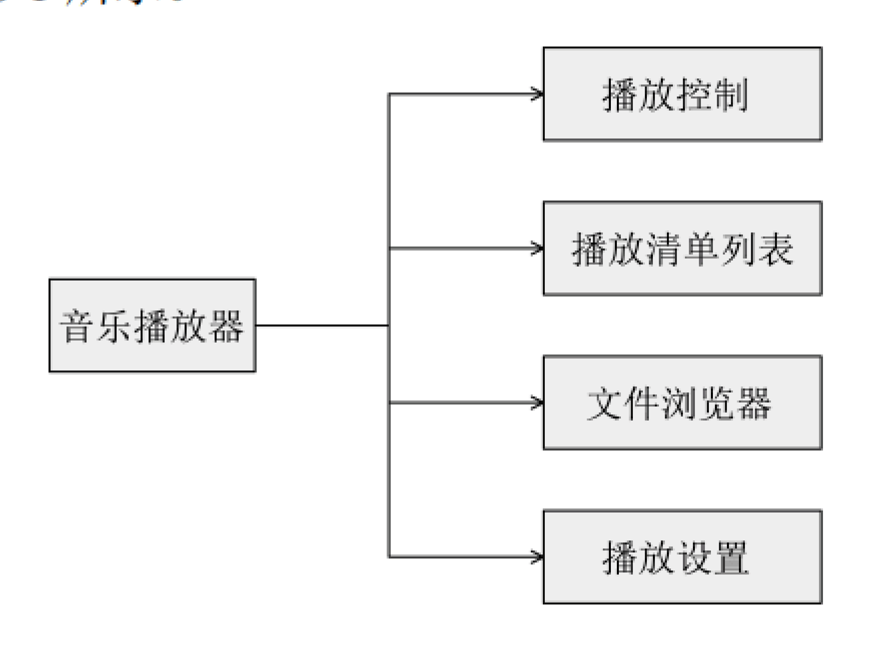


图 6-1 系统构成模块

各个模块的具体说明如下所示。

1. 播放控制模块

此模块的功能是控制播放的音乐文件。

（1）播放

☑ 目标：使得用户可以播放在播放列表中选中的歌曲。

☑ 前置条件：播放器正在运行。

☑ 基本事件流。

  ➢ 用户单击“播放”按钮。

  ➢ 播放器将播放列表中当前的歌曲。

（2）暂停播放

☑ 目标：使得用户可以暂停正在播放的歌曲。

☑ 前置条件：歌曲正在播放且未停止和暂停。

☑ 基本事件流。

- ➢ 单击“暂停”按钮。
- ➢ 播放器将暂停当前的歌曲。

（3）停止播放

- ☑ 目标：使得用户可以停止正在播放的歌曲。
- ☑ 前置条件：歌曲正在播放或暂停。
- ☑ 基本事件流。
  - ➢ 用户单击“停止”按钮。
  - ➢ 播放器将停止当前播放的歌曲。

（4）上一首/下一首

- ☑ 目标：使得用户可以听上一首或下一首歌曲。
- ☑ 前置条件：歌曲正在播放或暂停。
- ☑ 基本事件流。
  - ➢ 用户单击“上一首”或“下一首”按钮。
  - ➢ 播放器将播放上一首或下一首歌曲。

（5）播放清单

- ☑ 目标：使得用户可以进入播放清单。
- ☑ 前置条件：程序在运行。
- ☑ 基本事件流。
  - ➢ 用户单击“清单”按钮。
  - ➢ 播放器进入清单列表。

播放控制的基本结构如图 6-2 所示。

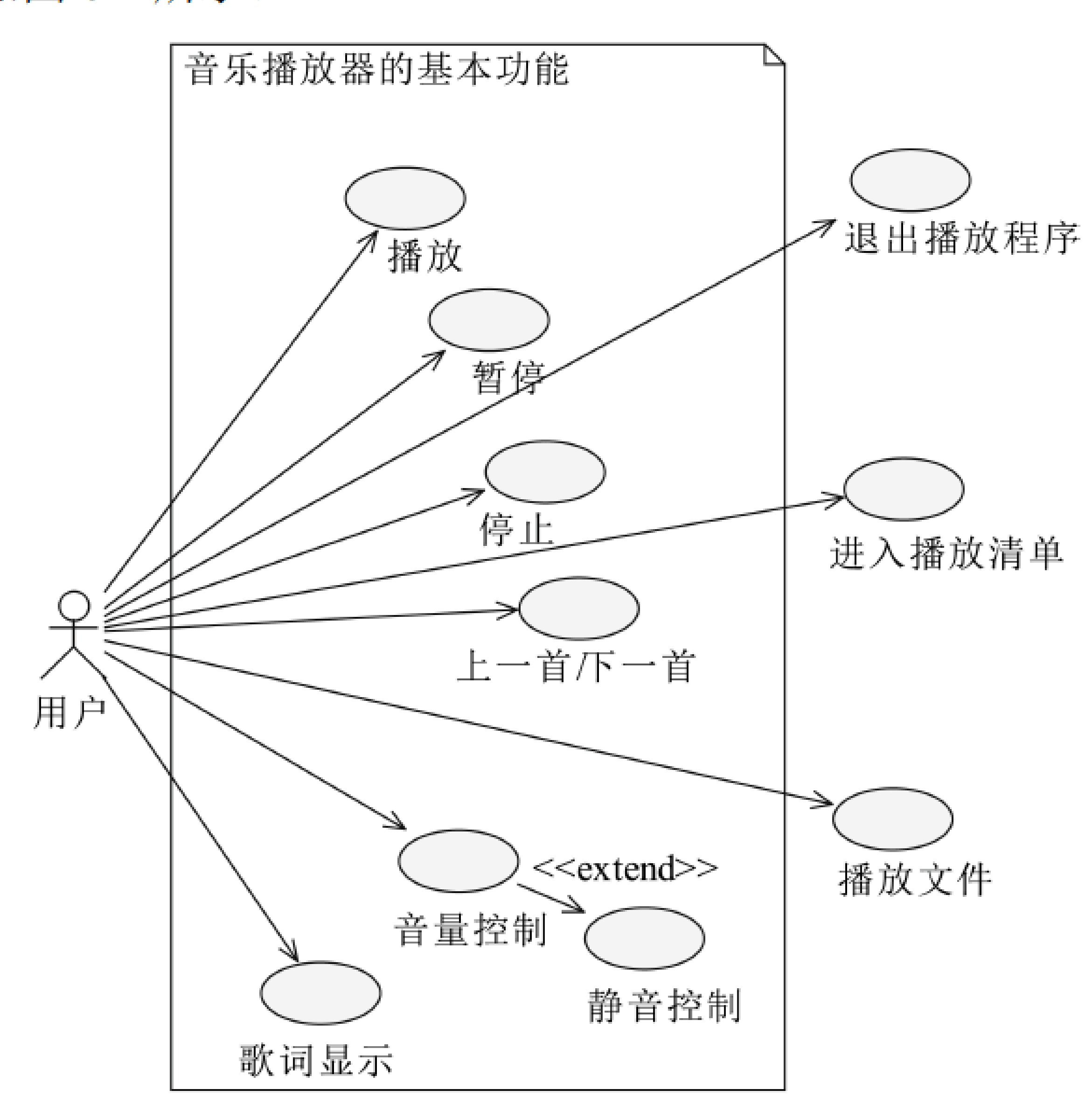


图 6-2　播放控制模块的结构

2. 播放清单列表模块

当用户选中列表中的某一首歌曲后会显示播放清单，可以进行如下操作。

（1）播放

☑ 目标：使程序播放选中的歌曲。
☑ 前置条件：程序运行在播放菜单选项中。
☑ 基本事件流。
  ➢ 用户单击“播放”按钮。
  ➢ 播放器进入播放状态。

（2）视频详情

☑ 目标：使得程序显示歌曲详情。
☑ 前置条件：程序运行在播放菜单选项中。
☑ 基本事件流。
  ➢ 用户单击“详细”按钮。
  ➢ 显示歌曲详细状态。

（3）增加

☑ 目标：使得程序进入手机扩展 SD 卡。
☑ 前置条件：程序运行在播放菜单选项中。
☑ 基本事件流。
  ➢ 用户单击“增加”按钮。
  ➢ 播放器进入手机扩展 SD 卡。

（4）移除/全部移除

☑ 目标：使选中的歌曲被移除。
☑ 前置条件：程序运行在播放菜单选项中。
☑ 基本事件流。
  ➢ 用户单击“移除/全部移除”按钮。
  ➢ 播放器移除选中歌曲/全部移除歌曲。

（5）设定

☑ 目标：使得程序进入播放器设定状态。
☑ 前置条件：程序运行在播放菜单选项中。
☑ 基本事件流。
  ➢ 用户单击“设定”按钮。
  ➢ 播放器进入设定界面。

本实例播放清单界面的结构如图 6-3 所示。

3. 播放设置模块

本模块用于设置音乐的播放方式，并设置是否显示歌词。

（1）播放模式

☑ 目标：使得程序进入播放模式设定状态。
☑ 前置条件：程序运行在播放器设定界面中。
☑ 基本事件流。
  ➢ 用户单击“顺序”“随机”“单曲”按钮。
  ➢ 播放器进入选中模式播放状态。

（2）显示歌词

- ☑ 目标：使得程序进入播放器歌词设置状态。
- ☑ 前置条件：程序运行在播放设定界面。
- ☑ 基本事件流。
  - ➢ 用户单击“歌词开关”按钮。
  - ➢ 播放器显示或关闭歌词。

本模块设置的结构如图 6-4 所示。

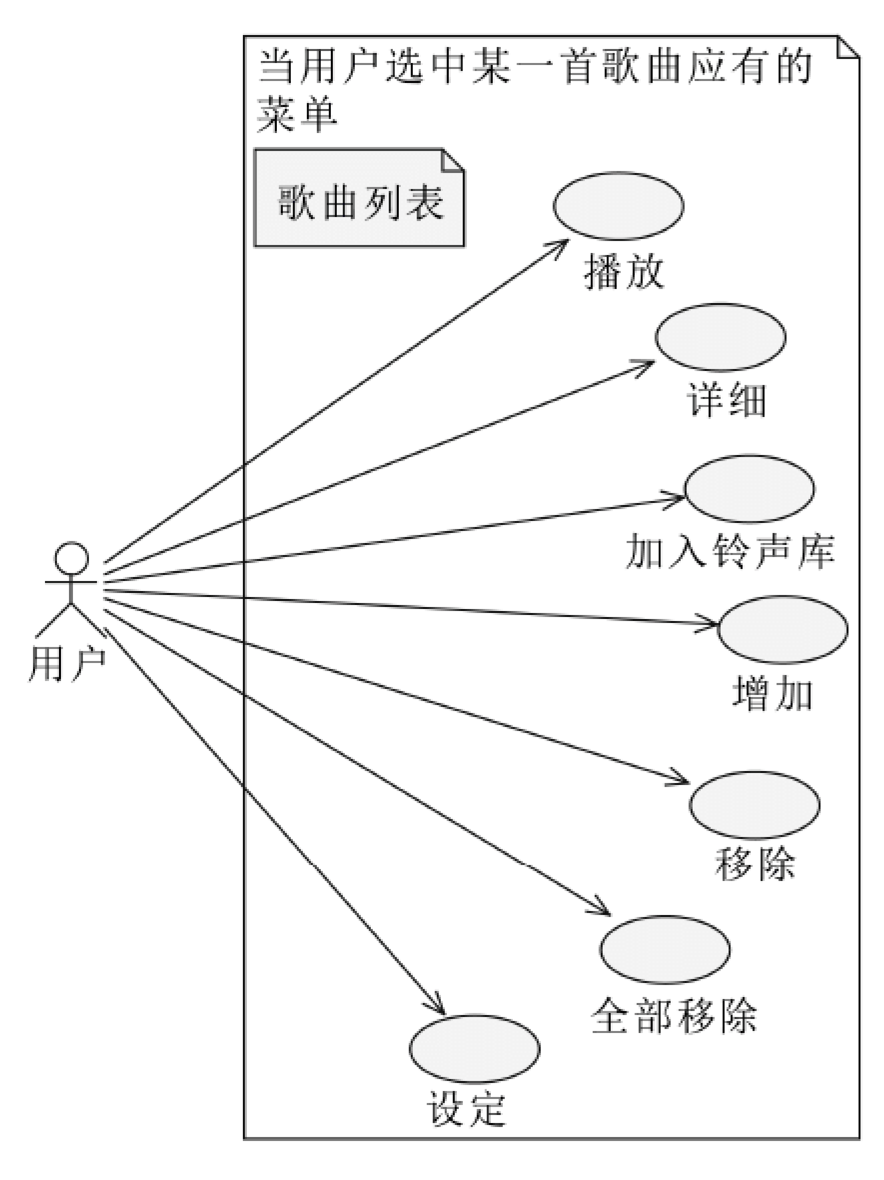


图 6-3　播放清单界面结构

播放设定
单曲循环
顺序播放
随机播放
播放模式
用户
系统设定
歌词显示
歌词开启
歌词关闭

图 6-4　设置界面结构

4. 文件浏览器模块

此模块的功能是浏览系统内或 SD 卡中的文件信息。

（1）SDcard

- ☑ 目标：使得程序进入 SDcard 目录。
- ☑ 前置条件：程序运行目录界面。
- ☑ 基本事件流。
  - ➢ 用户单击 SDcard 选项。
  - ➢ 程序进入 SDcard 目录。

（2）System

- ☑ 目标：使得程序进入 System 目录。
- ☑ 前置条件：程序运行目录界面。
- ☑ 基本事件流。
  - ➢ 用户单击 System 选项。
  - ➢ 程序进入 System 目录下。

文件浏览器模块的结构如图 6-5 所示。

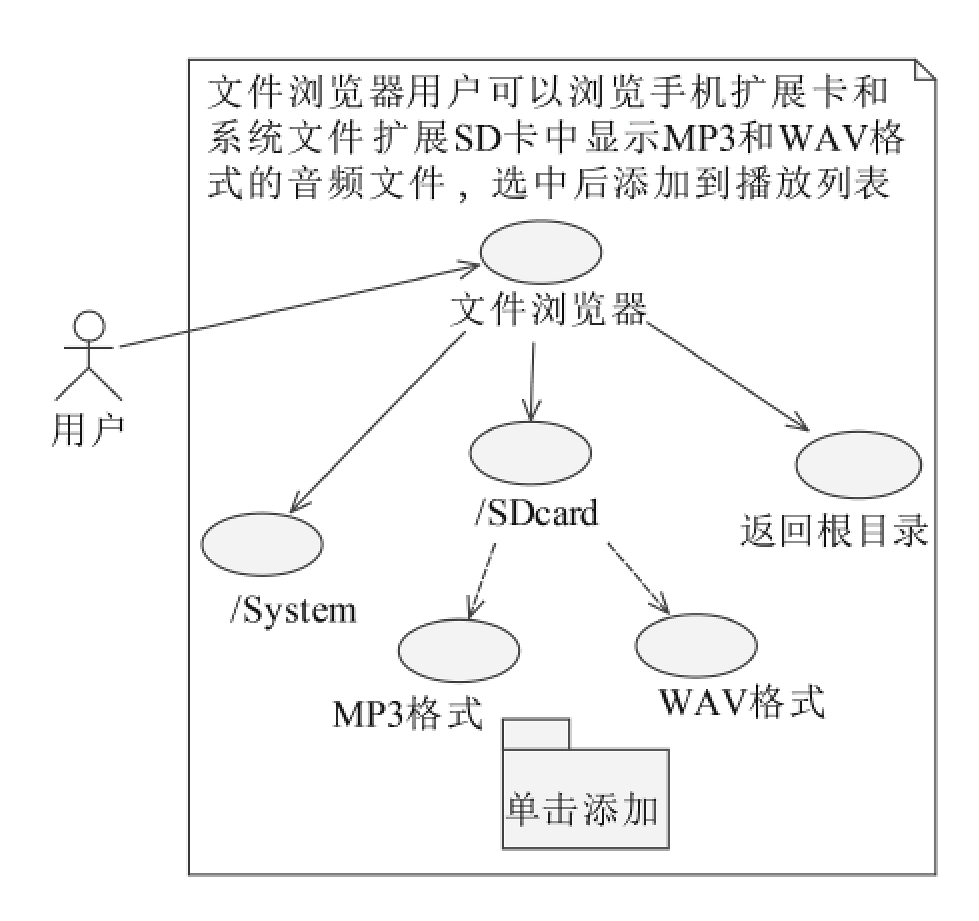


图 6-5　文件浏览器模块结构

6.2.2　系统流程

本项目音乐播放器的系统流程如图 6-6 所示。

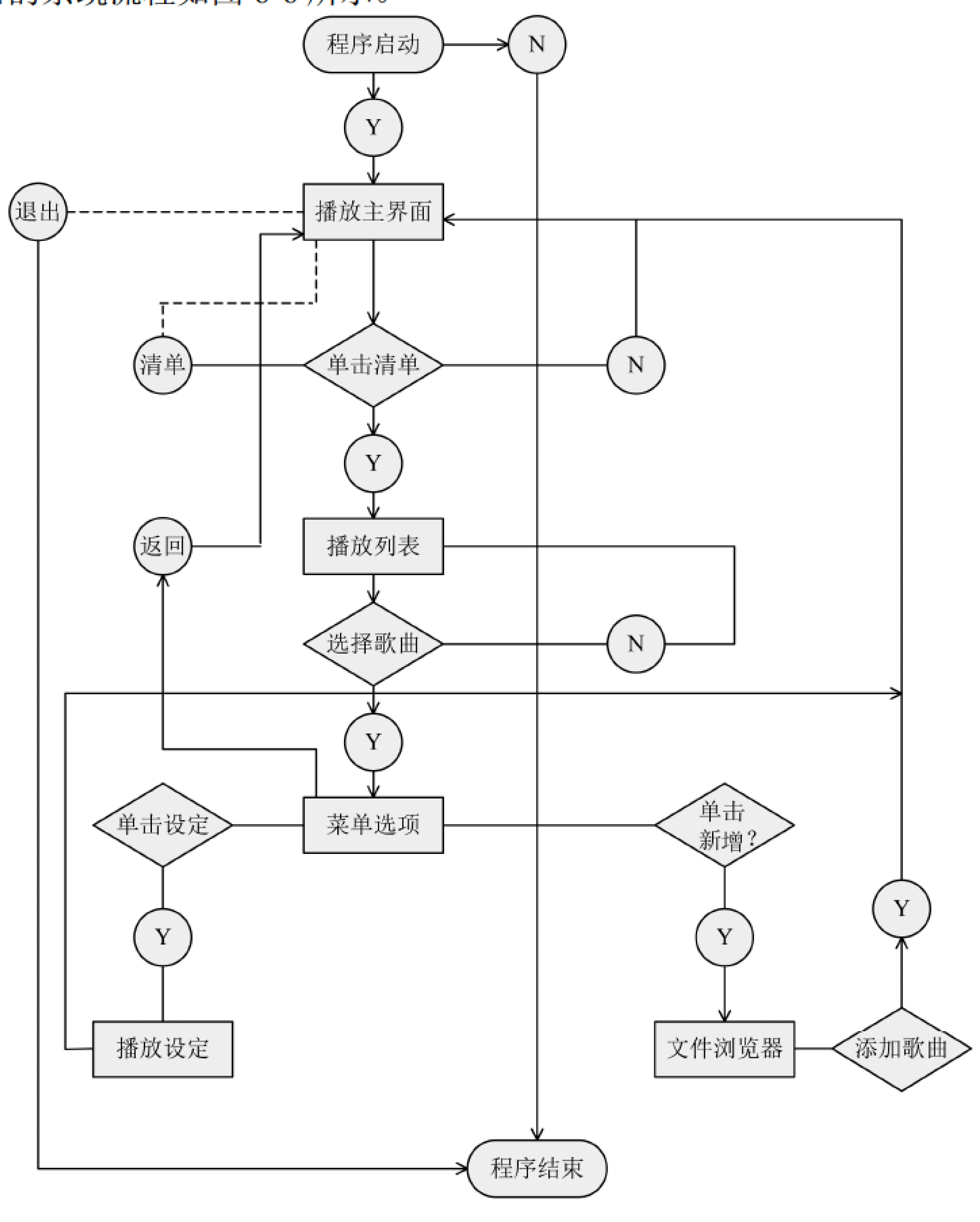


图 6-6　音乐播放器系统流程图

6.2.3　功能结构图

本项目音乐播放器系统的完整功能结构如图 6-7 所示。

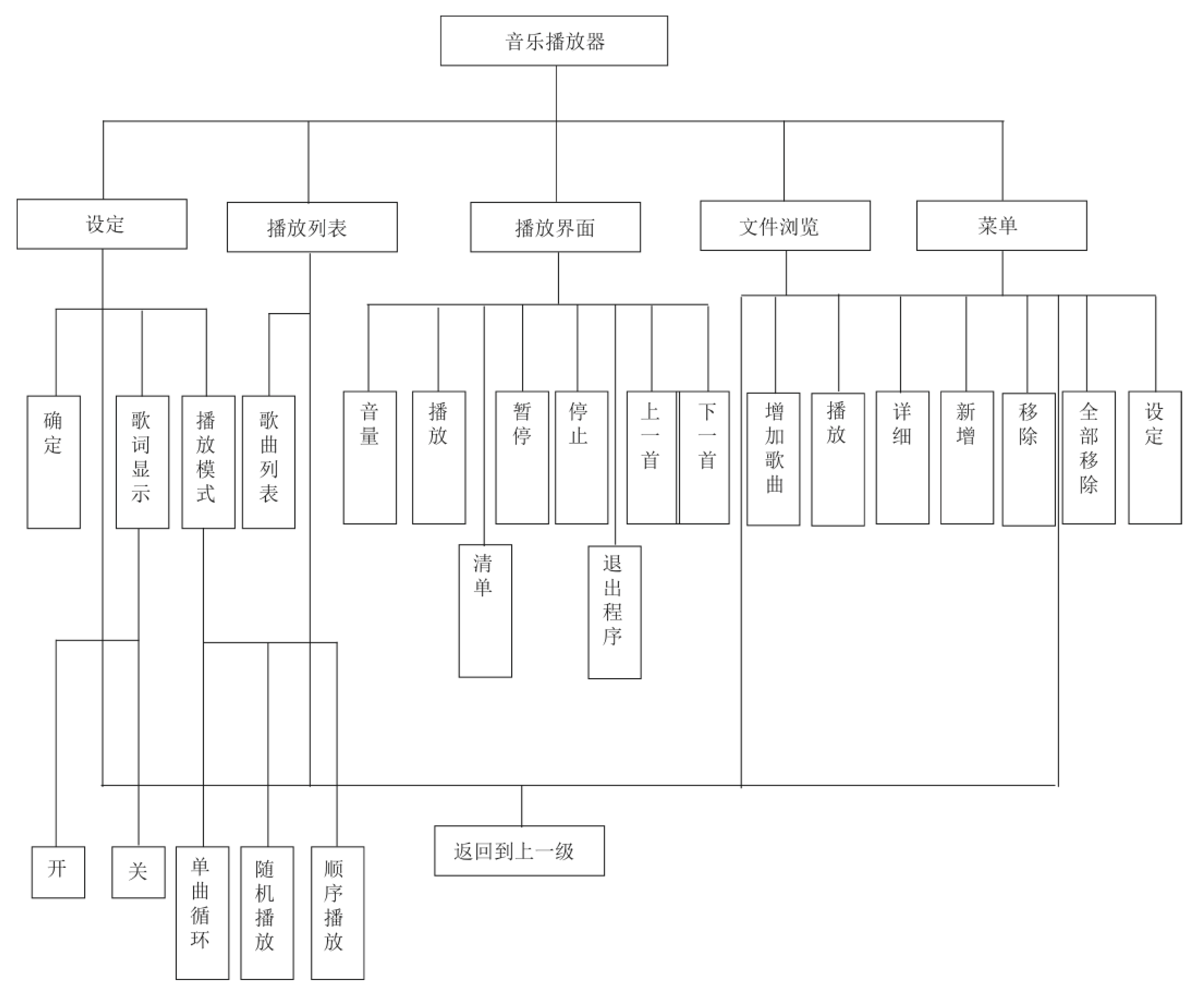


图 6-7　完整功能结构图

6.2.4　系统功能说明

本项目音乐播放器系统各个模块功能的说明如表 6-1 所示。

表 6-1　模块结构功能说明信息表

功 能 类 别	子　功　能	子　功　能
播放列表	播放列表菜单	退出播放
		从扩展卡寻找歌曲
	歌曲菜单	播放→进入播放界面
		删除→数据库同步更新
		重命名→数据库同步更新
		向上、下移动→数据库同步更新

续表

功 能 类 别	子　功　能	子　功　能
播放界面	播放	播放歌曲→线程启动→时间更新
	暂停	暂停歌曲→线程暂停→时间暂停
	停止	停止歌曲→线程停止→时间停止
	上一首	播放列表索引变化→寻找上一 ID 歌曲
	下一首	播放列表索引变化→寻找下一 ID 歌曲
	播放界面菜单	返回到播放列表
		返回到主菜单
		从扩展卡寻找歌曲
		退出播放器
		隐藏播放界面
主菜单	退出程序	程序退出
	进入播放列表	显示播放列表

6.2.5　系统需求

（1）系统界面需求

播放器界面要求布局合理、颜色舒适、控制按钮友好，为了减少开发工作量，图片素材多数为公司项目素材，如图 6-8 所示。

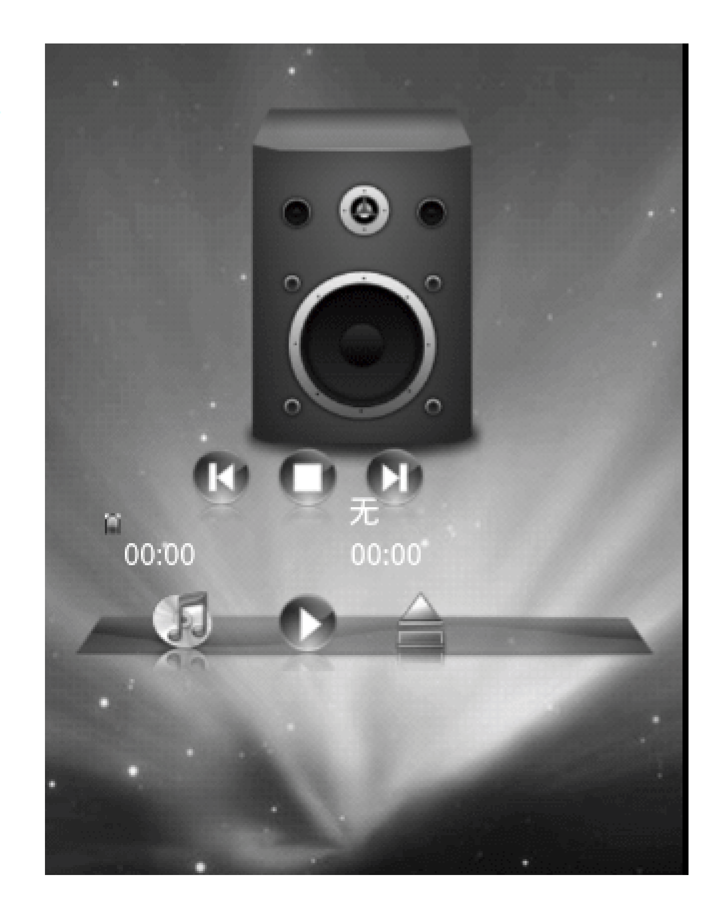


图 6-8　播放器主界面

（2）系统性能需求

根据 Android 手机系统要求无响应时间为 5 秒，所以有如下性能要求。

☑　当要求歌曲播放时，程序响应时间最长不能超过 5 秒。
☑　当要求歌曲暂停时，程序响应时间最长不能超过 5 秒。
☑　当要求歌曲停止时，程序响应时间最长不能超过 5 秒。
☑　当要求歌曲上/下一首时，程序响应时间最长不能超过 5 秒。
☑　当要求进行清单列表时，程序响应时间最长不能超过 5 秒。

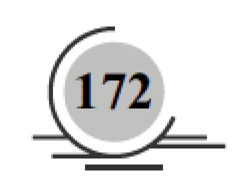
（3）运行环境需求

☑　操作系统：Android 手机基于 Linux 操作系统。
☑　支持环境：Android 1.5 - 2.0.1 版本。
☑　开发环境：Eclipse 3.5 ADT 0.95。

6.3　数据库设计

知识点讲解：光盘:视频\视频讲解\第 6 章\数据库设计.avi

数据库是存放数据的仓库，只不过这个仓库是在计算机存储设备上，而且数据是按一定的格式存放的。数据库中的数据按一定数据模型组织、描述和存储，具有较小的重复度、较高的数据独立性和易扩展性，并且可以在一定范围内被各种用户共享。在涉及数据库的软件开发中，需要根据有待解决的问题性质、规模，以及所采用的前端程序创建工具等，做出合适的数据库类型选择。



6.3.1　字段设计

字段 file_table 用于保存歌曲的名字、类型和路径，具体说明如表 6-2 所示。

表 6-2　字段 file_table 说明

属　　性	数 据 类 型	说　　明
Id	INTEGER	id 号
fileName	TEXT	歌曲名字
filePath	TEXT	歌曲路径
sort	INTEGER	歌曲类型

SD 卡中保存歌曲用表 6-3 中的字段来保存。

表 6-3　歌曲详情表

属　　性	数 据 类 型	说　　明
ID	INTEGER	id 号
TITLE	TEXT	标题
ARTIST	TEXT	艺术家
ALBUM	TEXT	专辑
SIZE	LONG	大小

在 Android 系统中，通过自带的 MediaStore 封闭类来存储媒体信息，通过 Uri EXTERNAL_CONTENT_URI 来访问 SDcard 中的歌曲详细信息。

6.3.2　E-R 图设计

音乐播放器的 E-R 图如图 6-9 所示。

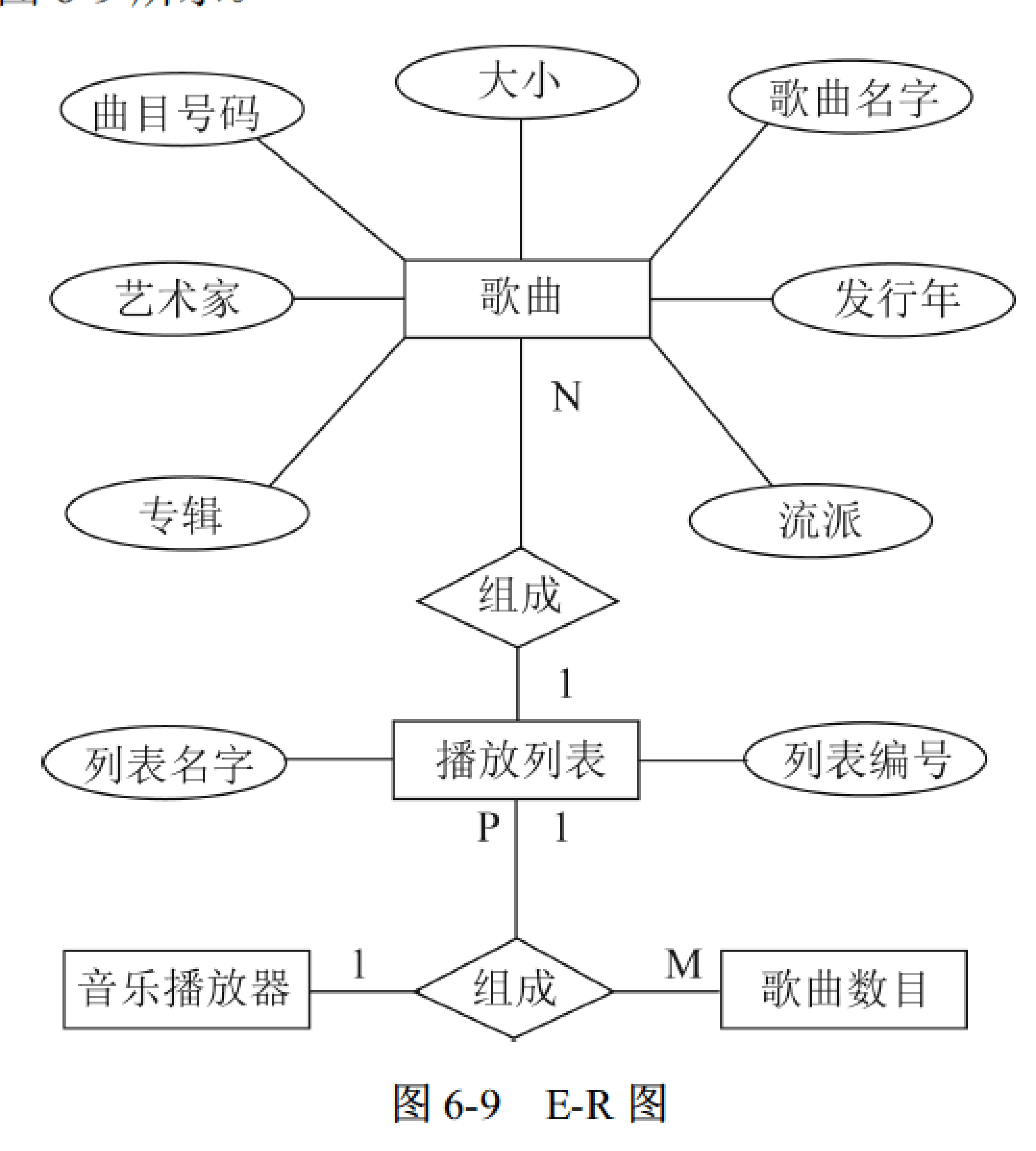


图 6-9　E-R 图

6.3.3 数据库连接

Android 系统中自带的 iSQLite 数据库是一个十分小型的数据库，这样正适合 Android 这种移动平台使用。Android 数据库存储的位置在 data/data/<项目文件夹>/databases/目录下。

Android 使用 ContentProvider 作为内容提供商，SQLiteOpenHelper 数据库帮助类来进行对数据库的创建和操作。通过 Context.getContentResolver()方法直接对数据库进行操作。程序中数据库类为 DBHelper extends SQLiteOpenHelper（继承关系），内容提供类为 DBProvider extends ContentProvider（继承关系）。

6.3.4 创建数据库

Android 提供了标准的数据库创建方式，继承自 SQLiteOpenHelper，实现 onCreate()和 onUpgrade() 两个方法，这样做的好处是便于数据库版本的升级。在此编写文件 DBHelper.java，实现连接数据库的算法，具体代码如下所示。

```
public class DBHelper extends SQLiteOpenHelper{
    /**
     * 数据库名称常量
     */
    private static final String DATABASE_NAME = "MyMusic.db";
    /**
     *  数据库版本常量
     */
    private static final int DATABASE_VERSION = 1;
    /**
     *  表名称常量
     */
    public static final String TABLES_TABLE_NAME = "File_Table";
    private static final String DATABASE_CREATE = "CREATE TABLE " + FileColumn.TABLE +" ("
     + FileColumn.ID+" integer primary key autoincrement,"
     + FileColumn.NAME+" text,"
     + FileColumn.PATH+" text,"
     + FileColumn.SORT+" integer,"
     + FileColumn.TYPE+" text)";
     /**
      *  构造方法
      * @param context
      */
     public DBHelper(Context context) {
         // 创建数据库
         super(context, DATABASE_NAME,null, DATABASE_VERSION);
     }
     /**
      *  创建时调用
      */
     public void onCreate(SQLiteDatabase db) {
         /*Locale l = new Locale("zh", "CN");
         db.setLocale(l);*/
         db.execSQL(DATABASE_CREATE);
     }
```

```
    /**
     *  版本更新时调用
     */
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // 删除表
        db.execSQL("DROP TABLE IF EXISTS File_Table");
       onCreate(db);
    }
}
```

如果创建数据库不成功，则抛出 FIleNotFoundException 异常。

6.3.5　操作数据库

Android 对数据库的操作主要有插入、删除、更新和查询操作，在进行任何操作时都必须指定一个 URI，才能对相应的表进行数据操作。编写文件 DBProvider.java，在其中分别编写数据插入、修改、查询和删除操作的实现方法，具体代码如下所示。

```
public class DBProvider extends ContentProvider {
    private DBHelper dbOpenHelper;
    public static final String AUTHORITY = "MUSIC";
    public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY
            + "/" + FileColumn.TABLE);
    @Override
    public int delete(Uri arg0, String arg1, String[ ] arg2) {
        SQLiteDatabase db = dbOpenHelper.getWritableDatabase();

        try {
            db.delete(FileColumn.TABLE, arg1, arg2);
            Log.i("info", "delete");
        } catch (Exception ex) {
            ex.printStackTrace();
            Log.e("error", "delete");
        }
        return 1;
    }
    /**
     * 待实现
     */
    @Override
    public String getType(Uri uri) {
        return null;
    }
    /**
     * 插入
     */
    @Override
    public Uri insert(Uri uri, ContentValues values) {
        SQLiteDatabase db = dbOpenHelper.getWritableDatabase();
        long count = 0;
        try {
```

```
                    count = db.insert(FileColumn.TABLE, null, values);
            } catch (Exception ex) {
                    ex.printStackTrace();
                    Log.e("error", "insert");
            }
            if (count > 0)
                    return uri;
            else
                    return null;
    }
    @Override
    public boolean onCreate() {
            dbOpenHelper = new DBHelper(getContext());
            return true;
    }
    /**
      * 根据条件查询
      * @return 数据集
      */
    @Override
    public int update(Uri uri, ContentValues values, String selection,
                    String[ ] selectionArgs) {
            SQLiteDatabase db = dbOpenHelper.getWritableDatabase();
            int i = 0;
            try {
                    i = db.update(FileColumn.TABLE, values, selection, null);
                    return i;
            } catch (Exception ex) {
                    Log.e("error", "update");
            }
            return 0;
    }
}
```

6.3.6 数据显示

本项目在显示数据时，利用 Cursor 游标类指向数据表中的某一项，然后进行查询数据，并用 Log 日志显示出来。

```
//数据库查询操作
@Override
    public Cursor query(Uri uri, String[ ] projection, String selection,String[ ] selectionArgs, String sortOrder) {
            SQLiteDatabase db = dbOpenHelper.getWritableDatabase();
//参数依次为：表名，查询字段，where 语句，替换，
//group by（分组）, having（分组条件）,order by（排序）
            Cursor cur = db.query(FileColumn.TABLE, projection, selection,selectionArgs, null, null, sortOrder);
return cur;}
```

6.4　具体编码

知识点讲解：光盘:视频\视频讲解\第 6 章\具体编码.avi

经过前面内容的讲解，本播放器实例项目的前期工作已经结束。在接下来的内容中，将详细讲解本项目的具体编码过程。

6.4.1　设置服务信息

编写文件 SystemService.java，在此设置项目的服务信息，主要代码如下所示。

```
public class SystemService {
	private Context context;
	private Cursor cursor;
	public SystemService(Context context) {
		this.context = context;
	}

	public Cursor allSongs() {
		if (cursor != null)
			return cursor;
		ContentResolver resolver = context.getContentResolver();
		cursor = resolver.query(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
				null, null, null, MediaStore.Audio.Media.DEFAULT_SORT_ORDER);
		return cursor;
	}

	/**
	 * 读取正在播放歌曲的艺术家
	 * @return
	 */
	public String getArtist() {
		return cursor.getString(cursor
				.getColumnIndexOrThrow(MediaStore.Audio.Media.ARTIST));
	}
	/**
	 * 读取正在播放歌曲名字
	 * @return 歌曲名字
	 */
	public String getTitle() {
		String title = cursor.getString(cursor
				.getColumnIndexOrThrow(MediaStore.Audio.Media.TITLE));
		try {
			title=EncodingUtils.getString(title.getBytes(), "UTF-8");
		} catch (Exception e) {

			e.printStackTrace();
		}
		return title;
	}
```

```
/**
 *   读取正在播放歌曲的专辑
 * @return  专辑名
 * @throws RemoteException
 */
public String getAlbum() throws RemoteException {
        return cursor.getString(cursor
                        .getColumnIndexOrThrow(MediaStore.Audio.Media.ALBUM));
}

/*public int getDuration() throws RemoteException {
        //获得当前歌曲的时长
        return player.getDuration();
}public int getTime() throws RemoteException {
        //获得当前的媒体时间
        return player.getCurrentPosition();
}
```

6.4.2 播放器主界面

Android 的每一个可视化界面，都有其唯一的布局配置文件，该文件中有各种布局方式和各种资源文件，如图像、文字、颜色的引用，程序在运行时，可以通过代码对各配置文件进行读取。这样就可以形成不同的可视化界面和炫丽的效果。

（1）本实例主界面的布局文件是 main.xml，主要代码如下所示。

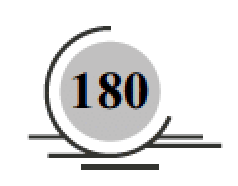
```
<TextView
       android:id="@+id/current_music"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:textSize="16sp"
       android:textColor="#ffffff"
        android:padding="10dip"
        android:cacheColorHint="#00000000"
       android:text="this is TextView…"
       android:layout_y="320px"/>
   -->
<Gallery android:id="@+id/gallery"
            android:layout_width="fill_parent"
                    android:layout_height="200dp"
                    android:layout_alignParentLeft="true"
                    android:spacing="16dp"
                    android:cacheColorHint="#00000000"
                    android:layout_centerVertical="true"
                    android:layout_y="20px"/>
<SeekBar android:id="@+id/seekbar" android:layout_width="245px"
             android:layout_height="20px" android:layout_x="40px"
        android:progressDrawable="@drawable/seekbar_style"
         android:thumb="@drawable/thumb"
             android:paddingLeft="18px"
```

```
            android:paddingRight="15px"
            android:paddingTop="5px"
            android:paddingBottom="5px"
            android:progress="0"
            android:max="100"
            android:secondaryProgress="0"
            android:layout_y="350px"/>
<TextView android:layout_x="60px" android:layout_height="wrap_content"
             android:text="00:00" android:layout_y="370px" android:id="@+id/current_time_text"
             android:layout_width="wrap_content"></TextView>
      <TextView android:layout_x="230px" android:layout_height="wrap_content"
            android:text="00:00" android:layout_y="370px" android:id="@+id/end_Time_Text"
            android:layout_width="wrap_content"></TextView>
<LinearLayout android:orientation="horizontal"
             android:gravity="center"
             android:layout_y="423px"
             android:layout_height="wrap_content"
             android:layout_width="fill_parent"
             android:background="@drawable/buttonground" />
<!-- 建立第 1 个 ImageButton -->
  <ImageButton
       android:id="@+id/btStart"
       android:layout_height="70dp"
       android:layout_width="70dp"
       android:layout_x="145px"
       android:layout_y="390px"
       android:background="#00000000"
       android:src="@drawable/play"
  />
<!-- 建立第 2 个 ImageButton-->
  <ImageButton
       android:id="@+id/pause"
       android:layout_height="wrap_content"
       android:layout_width="wrap_content"
       android:background="#00000000"
       android:layout_x="141px"
       android:layout_y="50px"
       android:src="@drawable/pause"
  />
<!-- 建立第 3 个 ImageButton -->
  <ImageButton
       android:id="@+id/before"
       android:layout_height="70dp"
       android:layout_width="70dp"
       android:background="#00000000"
       android:layout_x="80px"
       android:layout_y="280px"
       android:src="@drawable/backward"
  />
<!-- 建立第 4 个 ImageButton -->
  <ImageButton
```

```
        android:id="@+id/next"
        android:layout_height="70dp"
        android:layout_width="70dp"
        android:background="#00000000"
        android:layout_x="210px"
        android:layout_y="280px"
        android:src="@drawable/forward"
    />
<!-- 建立第 5 个 ImageButton -->
        <ImageButton
        android:id="@+id/btStop"
        android:layout_height="70dp"
        android:layout_width="70dp"
        android:layout_x="145px"
        android:layout_y="280px"
        android:background="#00000000"
        android:src="@drawable/stop"
    />

    <ImageButton
        android:id="@+id/listplay"
        android:layout_height="70dp"
        android:layout_width="70dp"
        android:cacheColorHint="#00000000"
        android:layout_x="50px"
        android:layout_y="390px"
        android:background="#00000000"
        android:src="@drawable/itunes2" />
<!--
    <ImageButton
        android:id="@+id/player"
        android:layout_height="70dp"
        android:layout_width="70dp"
        android:cacheColorHint="#00000000"
        android:layout_x="140px"
        android:layout_y="390px"
        android:background="#00000000"
        android:src="@drawable/wmp2" />
     -->
    <ImageButton
        android:id="@+id/returnBt"
        android:layout_height="70dp"
        android:layout_width="70dp"
        android:cacheColorHint="#00000000"
        android:layout_x="230px"
        android:layout_y="390px"
        android:background="#00000000"
        android:src="@drawable/white"
/>
```

（2）播放器主界面是一个 Activity，Android 工程在每个 Activity 启动时会首先执行 Oncreate()方法。本实例主界面的程序文件是 MainPlayActivity.java，其具体实现流程如下所示。



☑ 实现界面初始化工作，如果有播放的歌曲则在播放器中显示歌曲名，并显示上次的播放进度。如果设置了显示歌词，则还会在界面中显示歌词。对应代码如下所示。

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.main);
        systemProvider=new SystemService(this);
        cursor=systemProvider.allSongs();
        SharedPreferences sp = getSharedPreferences("MUSIC",MODE_WORLD_READABLE);
        if (sp != null) {
            playingName = sp.getString("PLAYINGNAME", null);
            selectName = sp.getString("SELECTNAME", null);
            String s = sp.getString("MUSIC_LIST", null);
            if (s != null)
                music_List = StringHelper.spiltString(s);
        }

        init_Play_Rack();//界面初始化
        if (playingName != null) {//
            int time1 = mplayer.getDuration();
            int time2 = mplayer.getCurrentPosition();
            seekBar.setMax(time1);
            seekBar.setProgress(time2);
            currently_Time.setText(getFileTime(time2));
            end_Time.setText(getFileTime(time1));
            currently_Music.setText(playingName);


            handler.removeCallbacks(thread_One);
            handler.postDelayed(thread_One, 1000);
            lrc_time = new ArrayList<String>();
            lrc_word = new ArrayList<String>();
            showLrc(playingName);//歌词显示
        }
        if (selectName != null) {//播放选中的歌曲
            play_bt.setImageBitmap(musicAdapter.getSuspend_Icon());//默认暂停图标
            play_Music();
            lrc_time = new ArrayList<String>();
            lrc_word = new ArrayList<String>();
            showLrc(selectName);//歌词显示
        }
        if (!(currently_Music.getText().toString()).equals("无")) {
            play_bt.setOnTouchListener(playListener);                    //播放监听器
            seekBar.setOnSeekBarChangeListener(seekBarListener);        //音轨监听器
            stop_bt.setOnTouchListener(stopListener);                    //停止监听器
            move_Down.setOnTouchListener(downListener);                 //下一首歌曲监听器
            move_Up.setOnTouchListener(upListener);                      //上一首歌曲监听器
        }
        list_bt.setOnTouchListener(list_bt_listener);                    //清单监听器
        back_bt.setOnTouchListener(return_bt_listener);
```

```
        mplayer.setOnCompletionListener(playerListener);        //监听歌曲是否播放完

        mSwitcher = (ImageSwitcher) findViewById(R.id.switcher);
        mSwitcher.setFactory(this);
        mSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
                android.R.anim.fade_in));
        mSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
                android.R.anim.fade_out));
        mSwitcher.setImageResource(R.drawable.background);
        Gallery g = (Gallery) findViewById(R.id.gallery);
        g.setAdapter(new ImageAdapter(this));
        g.setSelection(200);
        g.setOnItemSelectedListener(this);
    }
```

☑ 设置暂停和重置处理，对应代码如下所示。

```
protected void onPause() {
        super.onPause();
        SharedPreferences sp = getSharedPreferences("MUSIC",
                MODE_WORLD_WRITEABLE);
        SharedPreferences.Editor editor = sp.edit();
        playingName = currently_Music.getText().toString();
        if (!playingName.equals("无"))
            editor.putString("PLAYINGNAME", playingName);
        editor.putString("SELECTNAME", selectName);
        editor.putString("MUSIC_LIST", StringHelper.toStringAll(music_List));
        editor.commit();
        handler.removeCallbacks(thread_One);
    }
    @Override
    protected void onResume() {
        super.onResume();
        systemProvider=new SystemService(this);
        cursor=systemProvider.allSongs();
        SharedPreferences sp = getSharedPreferences("MUSIC",MODE_WORLD_READABLE);
        if (sp != null) {
            playingName = sp.getString("PLAYINGNAME", null);
            selectName = sp.getString("SELECTNAME", null);
            String s = sp.getString("MUSIC_LIST", null);
            if (s != null)
                music_List = StringHelper.spiltString(s);
        }
        if (mplayer.isPlaying()) {
            handler.removeCallbacks(thread_One);
            handler.postDelayed(thread_One, 1000);
        }
        else
            handler.removeCallbacks(thread_One);
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
```

```
        Log.i("info", "onDestroy");
        new File("/data/data/com.Rain.music.activity/shared_prefs/MUSIC.xml")
                .delete();
        new File("/data/data/com.Rain.music.activity/shared_prefs/SET_MSG.xml")
                .delete();
        System.exit(0);
    };
```

☑ 定义方法 init_Play_Rack()，实现主界面的初始化处理，对应代码如下所示。

```
    private void init_Play_Rack() {
        list_bt = (ImageButton) findViewById(R.id.listplay);
        back_bt = (ImageButton) findViewById(R.id.returnBt);
        stop_bt = (ImageButton) findViewById(R.id.btStop);
        play_bt = (ImageButton) findViewById(R.id.btStart);
        move_Up = (ImageButton) findViewById(R.id.before);
        move_Down = (ImageButton) findViewById(R.id.next);
        end_Time = (TextView) findViewById(R.id.end_Time_Text);
        //title_Music = (TextView) findViewById(R.id.title_music);
        currently_Time = (TextView) findViewById(R.id.current_time_text);
        currently_Music = (TextView) findViewById(R.id.current_music);
        seekBar = (SeekBar) findViewById(R.id.seekbar);

        mplayer = MusicHelp.getMediaPlayer();
        musicAdapter = new MusicAdapter(this, music_List);
        handler = MusicHelp.getHandler();
        currently_Music.setText("无");
        currently_Music.setTextColor(Color.WHITE);
        currently_Time.setTextColor(Color.WHITE);
        end_Time.setTextColor(Color.WHITE);
        lrcTime = (TextView) findViewById(R.id.lrcText);
        SharedPreferences sp = getSharedPreferences("SET_MSG",
                MODE_WORLD_READABLE);
        if (sp != null) {
            if (sp.getString("sigle_Play", null) != null) {
                play_Mode = sp.getString("sigle_Play", null);
            }
            if (sp.getString("order_Play", null) != null) {
                play_Mode = sp.getString("order_Play", null);
            }
            if (sp.getString("random_Play", null) != null) {
                play_Mode = sp.getString("random_Play", null);
            }
            if (sp.getString("lyLrc", null) != null) {
                lrc_Show = sp.getString("lyLrc", null);
            }
            Log.i("info", "play_Mode=" + play_Mode);
            Log.i("info", "lrc_Show=" + lrc_Show);
        }
    }
```

☑ 定义方法 onCompletion()，实现歌曲播放完监听器功能，对应代码如下所示。

```
    OnCompletionListener playerListener = new OnCompletionListener() {
```

```
		@Override
		public void onCompletion(MediaPlayer mp) {

			play_Mode();//播放模式
			lrc_time = new ArrayList<String>();
			lrc_word = new ArrayList<String>();
			showLrc(selectName);
		}
	};
```

☑ 定义 return_bt_listener，实现结束监听器处理，对应代码如下所示。

```
	OnTouchListener return_bt_listener = new OnTouchListener() {
		@Override
		public boolean onTouch(View v, MotionEvent event) {
			if (event.getAction() == MotionEvent.ACTION_DOWN) {
				back_bt.setImageResource(R.drawable.whitepress);
			} else if (event.getAction() == MotionEvent.ACTION_UP) {
				back_bt.setImageResource(R.drawable.white);
				finish();
				onDestroy();
//	android.os.Process.killProcess(android.os.Process.myPid());
//获取 PID，目前获取自己的只有该 API
//否则从/proc 中枚举其他进程，不过要说明的是，结束其他进程不一定有权限，否则易出现错误
			//	System.exit(0);
			}
			return false;
		}
	};
```

☑ 定义 list_bt_listener，实现清单监听器的处理，对应代码如下所示。

```
	OnTouchListener list_bt_listener = new OnTouchListener() {
		@Override
		public boolean onTouch(View v, MotionEvent event) {
			if (event.getAction() == MotionEvent.ACTION_DOWN) {
			//	v.setBackgroundResource(R.drawable.share_pressed);
				list_bt.setImageResource(R.drawable.itunespress);
			} else if (event.getAction() == MotionEvent.ACTION_UP) {
				//v.setBackgroundResource(R.drawable.share);
				list_bt.setImageResource(R.drawable.itunes2);

				Intent intent = new Intent(MainPlayActivity.this,
						PlayListActivity.class);
				startActivityForResult(intent, 0);
			}
			return false;
		}
	}
```

☑ 定义 OnSeekBarChangeListener seekBarListener，实现音轨监听器处理操作，对应代码如下所示。

```
	private OnSeekBarChangeListener seekBarListener = new OnSeekBarChangeListener() {
		@Override
		public void onProgressChanged(SeekBar seekBar, int progress,
				boolean fromUser) {
```

```
                if (fromUser) {
                    mplayer.seekTo(progress);
                    currently_Time.setText(getFileTime(progress));
                }
            }
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }
        };
```

☑ 定义 playListener，实现播放监听器的操作处理，对应代码如下所示。

```
OnTouchListener playListener = new OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    if (mplayer.isPlaying()) {
                        play_bt.setImageResource(R.drawable.pausepress);
                    }
                    else {
                        play_bt.setImageResource(R.drawable.playpress);
                    }
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    if (mplayer.isPlaying()) {
                        mplayer.pause();//暂停
                        currently_Time.setText(currently_Time.getText().toString());
                        play_bt.setImageBitmap(musicAdapter.getPlay_Icon());
                        handler.removeCallbacks(thread_One);
                    }
                    else {
                        if (is_stopping) {
                            play_Music();
                            is_stopping = false;
                            play_bt.setImageBitmap(musicAdapter.getSuspend_Icon());
                        } else {
                            mplayer.start();
                            handler.postDelayed(thread_One, 1000);
                            play_bt.setImageBitmap(musicAdapter.getSuspend_Icon());
                        }
                    }
                }
                return false;
            }
        };
```

☑ 定义 stopListener，实现停止监听器的处理，对应代码如下所示。

```
OnTouchListener stopListener = new OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
```

```
                    stop_bt.setImageResource(R.drawable.stoppress);
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    stop_bt.setImageResource(R.drawable.stop);
                    mplayer.stop();
                    currently_Time.setText("00:00");
                    end_Time.setText("00:00");
                    play_bt.setImageBitmap(musicAdapter.getPlay_Icon());
                    handler.removeCallbacks(thread_One);
                    seekBar.setProgress(1);
                    is_stopping = true;
                    lrcTime.setText("无");
                }
                return false;
            }
        };
```

☑ 定义 downListener，实现下一首歌曲监听器的操作处理，对应代码如下所示。

```
OnTouchListener downListener = new OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            move_Down.setImageResource(R.drawable.forwardpress);

        } else if (event.getAction() == MotionEvent.ACTION_UP) {
            move_Down.setImageResource(R.drawable.forward);
            move_Down(currently_Music.getText().toString());
            lrc_time = new ArrayList<String>();
            lrc_word = new ArrayList<String>();
            showLrc(currently_Music.getText().toString());//歌词显示
        }
        return false;
    }
};
```

☑ 定义 upListener，实现上一首歌曲监听器的操作处理，对应代码如下所示。

```
OnTouchListener upListener = new OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            move_Up.setImageResource(R.drawable.backwardpress);
        } else if (event.getAction() == MotionEvent.ACTION_UP) {
            move_Up.setImageResource(R.drawable.backward);
            move_Up(currently_Music.getText().toString());
            lrc_time = new ArrayList<String>();
            lrc_word = new ArrayList<String>();
            showLrc(currently_Music.getText().toString());// 歌词显示
        }
        return false;
    }
};
```

☑ 定义方法 move_Down()和 move_Up()，分别用于播放下一首歌曲和上一首歌曲，对应代码如下所示。

```
    private void move_Down(String musicName) {
```

```
        for (int i = 0; i < music_List.size(); i++) {
            if (musicName.equals(music_List.get(i))) {
                if ((i + 1) < music_List.size()) {
                    selectName = music_List.get(i + 1);
                    play_Music();
                    return;
                } else {
                    selectName = music_List.get(0);
                    play_Music();
                    return;
                }
            }
        }
    }

    private void move_Up(String musicName) {
        for (int i = 0; i < music_List.size(); i++) {
            if (musicName.equals(music_List.get(i))) {
                if ((i - 1) >= 0) {
                    selectName = music_List.get(i - 1);//移动到上一首歌曲
                    play_Music();
                    return;
                } else {
                    selectName = music_List.get(music_List.size() - 1);
                    play_Music();
                    return;
                }
            }
        }
    }
```

☑ 定义方法 play_Mode()来设置系统的播放模式，本实例有单曲循环、顺序播放和随机播放 3 种模式。对应代码如下所示。

```
private void play_Mode() {
    if ("is_Sigle".equals(play_Mode)) {//单曲循环
        play_Music();
    }
    if ("is_Order".equals(play_Mode)) {//顺序播放
        move_Down(currently_Music.getText().toString());
    }
    if ("is_Random".equals(play_Mode)) {//随机播放
        Random r = new Random();
        int idx = r.nextInt(music_List.size());//随机生成 [0,music_List.size())的 INT 值
        selectName = music_List.get(idx);
        play_Music();
    }
}
```

☑ 定义方法 play_Music()来播放指定的音乐文件，对应代码如下所示。

```
private void play_Music() {
    try {
```

```
            mplayer.reset();
            mplayer.setDataSource(query());//文件流中选择歌曲
            mplayer.prepare();
            mplayer.start();
            currently_Music.setText(selectName);
            /*
            if(cursor.moveToFirst()){
                String title=systemProvider.getArtist();
                currently_Music.setText(title);
            }*/

            seekBar.setMax(mplayer.getDuration());//音频文件持续时间
            seekBar.setProgress(1);
            currently_Time.setText(getFileTime(mplayer.getCurrentPosition()));
    //      lrcTime.setText(systemProvider.getArtist());
            handler.removeCallbacks(thread_One);
            end_Time.setText(getFileTime(mplayer.getDuration()));
            handler.postDelayed(thread_One, 1000);
        } catch (Exception e) {
            e.printStackTrace();
        }
}
```

☑ 定义方法 query()来查询歌曲路径，对应代码如下所示。

```
public String query() {
    ContentResolver cr = getContentResolver();
    Uri uri = DBProvider.CONTENT_URI;
    String[ ] projection = { "path" };
    String selection = "fileName=?";
    String[ ] selectionArgs = { selectName };
    Cursor c = cr.query(uri, projection, selection, selectionArgs, null);
    if (c.moveToFirst()) {
        String path = c.getString(0);
        return path;
    }
    return null;
}
```

☑ 定义方法 getFileTime()来获取音乐文件的播放持续时间长的格式化字符串，其返回值是一个格式化时间字符串。对应部分代码如下所示。

```
private String getFileTime(int timeMs) {
    int totalSeconds = timeMs / 1000;// 获取文件有多少秒
    StringBuilder mFormatBuilder = new StringBuilder();
    Formatter mFormatter = new Formatter(mFormatBuilder, Locale
            .getDefault());
    int seconds = totalSeconds % 60;
    int minutes = (totalSeconds / 60) % 60;
    int hours = totalSeconds / 3600;
    mFormatBuilder.setLength(0);

    if (hours > 0) {
        return mFormatter.format("%d:%02d:%02d", hours, minutes, seconds)
```

```
                        .toString();//格式化字符串
        } else {
            return mFormatter.format("%02d:%02d", minutes, seconds).toString();
        }
```

另外，在文件 MainPlayActivityRoot.java 中定义了主界面中控制按钮的响应方法，其主要实现代码如下所示。

```
/**
 * 清单按钮
 */
protected ImageButton list_bt;
/**
 * 返回按钮
 */
protected ImageButton back_bt;
/**
 * 停止按钮
 */
protected ImageButton stop_bt;
/**
 * 播放按钮
 */
protected ImageButton play_bt;
/**
 * 上一首歌曲
 */
protected ImageButton move_Up;
/**
 * 下一首歌曲
 */
protected ImageButton move_Down;
/**
 * 歌曲结束时间
 */
protected TextView end_Time;
/**
 * 当前播放时间
 */
protected TextView currently_Time;
/**
 * 当前播放时间
 */
protected TextView currently_Music;
/**
 * 音轨
 */
protected SeekBar seekBar;
/**
 * 歌词显示
 */
protected TextView lrcTime;
```

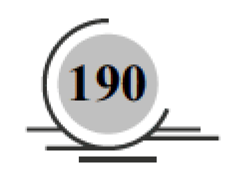
```
/**
 * 播放器是否停止
 */
protected boolean is_stopping = false;
/**
 * 系统自带播放器控件
 */
protected MediaPlayer mplayer;
/**
 * 选中的歌曲
 */
protected String selectName;
/**
 * 正在播放的歌曲
 */
protected String playingName;
/**
 *  播放模式 ：默认为随机播放模式
 */
protected String play_Mode = "is_Random";
/**
 * 歌词显示模式
 */
protected String lrc_Show;
/**
 * 歌曲列表
 */
protected List<String> music_List = new ArrayList<String>();
/**
 * 线程
 */
protected Handler handler;
```

6.4.3 播放列表功能

系统的歌曲列表界面如图 6-10 所示。



图 6-10 歌曲列表界面



（1）编写布局文件 play_list.xml，主要代码如下所示。

```
<LinearLayout android:layout_width="fill_parent"
    android:gravity="center" android:layout_height="wrap_content"
    android:background="@drawable/footer_bar">
    <TextView android:text="歌曲列表" android:id="@+id/music_list"
        android:textSize="@dimen/music_list_title" android:textStyle="bold"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"></TextView>

</LinearLayout>

<ListView android:id="@+id/show_play_list"
    android:layout_width="fill_parent" android:layout_height="337px"></ListView>
<LinearLayout android:layout_width="fill_parent"
    android:gravity="right" android:layout_height="wrap_content"
    android:background="@drawable/footer_bar">
    <ImageButton android:id="@+id/back" android:background="@drawable/back"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"></ImageButton>
</LinearLayout>
```

在 Android 系统中有一个名为 ListView 的视图，其特点是一个有 BaseAdapter 的属性，从下到下或从左到右的显示方式。系统默认的方式每一行只显示一个 TextView，本播放列表实现了自定义的方式，用 ListView 的每一行显示一个音乐图片和一个歌曲名字。在此定义了类 MusicAdapter 来继承 BaseAdapter，然后通过算法对这个适配器进行扩展，扩展成为第一行能显示一张图片和一个歌曲名字。由于 BaseAdapter 是一个抽象类，需要实现其中的抽象方法 getView()，该方法返回一个 View，即视图。视图可以显示在 Activity 上，所以就可以看到想要的歌曲列表界面。

ListView 同样有一个监听器 new onItemClickListener()，只要实现这个方法就可以监听鼠标的单击事件，当鼠标单击到每一行时，可以通过 ListView.getItemAtPositon(int position)得到该行上的信息。这样就可以通过 Intent 将数据传入到其他的 Activity。本程序的思路是当鼠标单击一行，会跳转到另一个 Activity 中，这个 Activity 和歌曲列表类似，也是一个 ListView，该界面将在下一节介绍。

歌曲列表是从播放主界面跳转过来的，能跳到该歌曲列表的前提是数据有歌曲列表的存在。因为每次歌曲列表显示时会查询数据库中的歌曲列表。如果不存在，则会提示是空列表，选择到 SDcard 中添加歌曲，如图 6-11 所示。



图 6-11　列表为空时的提示

（2）编写程序文件 PlayListActivity.java，首先定义方法 setListener()来监听用户的选择操作，将用户选择的歌曲进行播放，然后定义方法 query()来查询系统库内的歌曲，如果为空，则显示“播放列表为空”；最后定义方法 showDialog()来显示图 6-11 所示的提示框。文件 PlayListActivity.java 的主要代码如下所示。

```
private void setListener(){
    playlist.setOnItemClickListener(new OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                long arg3) {
            Intent intent = new Intent(PlayListActivity.this, Menu.class);
```

```
                selectName = playlist.getItemAtPosition(arg2).toString();
                startActivityForResult(intent, 2);
            }
        });
        back.setOnTouchListener(new OnTouchListener() {

            @Override
            public boolean onTouch(View v, MotionEvent event) {

                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    v.setBackgroundResource(R.drawable.back_pressed);
                } else if (event.getAction() == MotionEvent.ACTION_UP) {
                    v.setBackgroundResource(R.drawable.back);
                    Intent intent = new Intent();
                    intent.setClass(PlayListActivity.this, MainPlayActivity.class);
                setResult(0, intent);
                finish();
                }

                return false;
            }
        });
}

@Override
protected void onPause() {

        super.onPause();
        Log.v("log", "playListActivity is in pause state");
        SharedPreferences sp=getSharedPreferences("MUSIC", MODE_WORLD_WRITEABLE);
        SharedPreferences.Editor editor=sp.edit();


        editor.putString("SELECTNAME", selectName);
        String str=StringHelper.toStringAll(list);
        editor.putString("MUSIC_LIST", str);
        editor.commit();
}
public String[ ] query() {//查询数据库
        cr = getContentResolver();
        uri = DBProvider.CONTENT_URI;
        list.clear();
        String[ ] projection = { "filename", "path" };
        Cursor c = cr.query(uri, projection, null, null, null);
        if (c.getCount() == 0) {
            showDialog("播放列表为空");
        }
        String[ ] music = new String[c.getCount()];
        if (c.moveToFirst()) {
            for (int i = 0; i < c.getCount(); i++) {
                c.moveToPosition(i);
```

```
                String filename = c.getString(0);
                music[i] = filename;
                list.add(filename);
            }
        }
        if (music.length > 0) {
            playlist.setAdapter(new MusicAdapter(this, list));
        }
        return music;
}

private void showDialog(String msg) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage(msg).setCancelable(false).setPositiveButton("从 SD 卡",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        Intent intent = new Intent(PlayListActivity.this,
                                FileExplorerActivity.class);
                    //  startActivity(intent);
                        startActivityForResult(intent, 2);
                    }
                }).setNegativeButton("取消", new OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Intent intent = new Intent();
            setResult(0, intent);
            finish();
            }
        });
        AlertDialog alert = builder.create();
        alert.show();
}
```

6.4.4　菜单功能模块

本系统实例的菜单功能界面如图 6-12 所示。

图 6-12　菜单功能界面

（1）编写布局文件 menu.xml，主要代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
	android:orientation="vertical" android:layout_width="fill_parent"
	android:layout_height="fill_parent" android:background="@drawable/list_bg">
	<LinearLayout android:layout_width="fill_parent"
		android:gravity="center" android:layout_height="wrap_content"
		android:background="@drawable/footer_bar">
		<TextView android:text="选项" android:id="@+id/select_item"
			android:textSize="@dimen/music_list_title" android:textStyle="bold"
			android:layout_width="wrap_content"
			android:layout_height="wrap_content"></TextView>
	</LinearLayout>
	<ListView android:id="@+id/menu" android:layout_width="wrap_content"
		android:background="@drawable/list_item_bg"
		android:layout_height="wrap_content"></ListView>
	<TextView android:layout_width="wrap_content"
		android:layout_height="50px"></TextView>
	<LinearLayout android:layout_width="fill_parent"
		android:gravity="right"
		android:layout_height="wrap_content"
		android:background="@drawable/footer_bar">
		<ImageButton android:id="@+id/back"
			 android:background="@drawable/back"
			android:layout_width="wrap_content"
			android:layout_height="wrap_content"></ImageButton>
	</LinearLayout>
```

（2）编写文件 menu.java，其具体实现流程如下所示。

☑ 使用 List<String>容器来保存 String 类型的字符，保存了“播放”、“新增”、“详细”、“移除”、“全部移除”和“设置”等选项。对应部分代码如下所示。

```
protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	setContentView(R.layout.menu);
	menuLV = (ListView) findViewById(R.id.menu);
	back=(ImageButton)findViewById(R.id.back);
	select_Item = (TextView) findViewById(R.id.select_item);
	select_Item.setTextColor(Color.WHITE);
	SharedPreferences sp = getSharedPreferences("MUSIC",
			MODE_WORLD_READABLE);
	if (sp != null) {

		selectName=sp.getString("SELECTNAME", null);
	}

	List<String> seclect_items = new ArrayList<String>();
	seclect_items.add("播放");
	seclect_items.add("详细");
	seclect_items.add("新增");
	seclect_items.add("移除");
	seclect_items.add("全部移除");
	seclect_items.add("设置");
	menuLV.setAdapter(new MusicAdapter(this, seclect_items));
```

```
    back.setOnTouchListener(backListener)
    ;
```

☑ 使用 case 语句根据用户选择的选项来到对应的界面，对应部分代码如下所示。

```
    menuLV.setOnItemClickListener(new OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                long arg3) {
            SharedPreferences sp=getSharedPreferences("MUSIC", MODE_WORLD_WRITEABLE);
            SharedPreferences.Editor editor=sp.edit();
            switch (arg2) {
            case 0://播放
                Bundle bundle = new Bundle();
            bundle.putInt("operate", 0);
            Intent intent = new Intent();
            intent.putExtras(bundle);
            setResult(2, intent);
            finish();
                break;
            case 2:
                Intent intent_add = new Intent(Menu.this,
                        FileExplorerActivity.class);
                editor.putString("SELECTNAME", null);
                editor.commit();
            //  startActivity(intent_add);
                startActivityForResult(intent_add, 3);
                break;
            case 3://移除
                showDialog(selectName);
                break;
            case 4://全部移除
                showDialog("");
                break;
            case 5://设置
                Intent intent_set = new Intent(Menu.this, PlaySetting.class);
                editor.putString("SELECTNAME", null);
                editor.commit();
                startActivityForResult(intent_set, 3);
                break;
            default:
                break;
            }
        }
    });
}
```

☑ 定义方法 showDialog()，用于提示用户确认是否移除或全部移除列表中的音频文件。对应代码如下所示。

```
private void showDialog(final String msg) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    if (msg.equals(""))
        builder.setMessage("全部移除?");
```

```
        else
                builder.setMessage("是否移除?");
        builder.setCancelable(false).setPositiveButton("是",
                        new DialogInterface.OnClickListener() {
                                public void onClick(DialogInterface dialog, int id) {
                                        if (msg.equals(""))
                                                del_All();
                                        else
                                                del_One(selectName);
                                        Intent intent = new Intent();
                                setResult(6, intent);
                                finish();
                                }
                        }).setNegativeButton("否", new OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                }
        });
        AlertDialog alert = builder.create();
        alert.show();
}
```

☑ 定义方法 del_One()来删除单首歌曲，定义方法 del_All()来删除全部歌曲。对应代码如下所示。

```
private void del_One(String musicName) {//删除单首歌曲
        ContentResolver cr = getContentResolver();
        Uri uri = DBProvider.CONTENT_URI;
        String where = "fileName=?";
        String[ ] selectionArgs = { musicName };
        cr.delete(uri, where, selectionArgs);
}

private void del_All() {//删除全部歌曲
        ContentResolver cr = getContentResolver();
        Uri uri = DBProvider.CONTENT_URI;
        cr.delete(uri, null, null);
}
```

6.4.5 播放设置界面

本系统的播放设置界面如图 6-13 所示。



图 6-13 播放设置界面

（1）编写布局文件 setting.xml，主要代码如下所示。

```
<LinearLayout android:layout_width="fill_parent"
    android:gravity="center" android:layout_height="wrap_content"
    android:background="@drawable/footer_bar">
    <TextView android:text="设定" android:id="@+id/setting"
        android:textSize="@dimen/music_list_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"></TextView>
</LinearLayout>
<LinearLayout android:orientation="horizontal"
    android:layout_width="wrap_content"
    android:gravity="center_vertical"
    android:layout_height="wrap_content">
    <TextView android:text="播放模式" android:id="@+id/setting"
        android:textSize="@dimen/text_size"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"></TextView>
    <RadioGroup android:id="@+id/RadioGroup"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <RadioButton android:text="单曲循环" android:id="@+id/sigle_play"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"></RadioButton>
        <RadioButton android:text="顺序播放" android:id="@+id/order_play"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"></RadioButton>
        <RadioButton android:text="随机播放" android:id="@+id/random_play"
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"></RadioButton>
    </RadioGroup>
</LinearLayout>
<LinearLayout android:orientation="horizontal"
    android:layout_width="wrap_content" android:gravity="center_vertical"
    android:layout_height="wrap_content">
    <TextView android:text="歌词显示" android:id="@+id/setting"
        android:textSize="@dimen/text_size" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <ToggleButton android:text="" android:id="@+id/ly_lrc"
        android:checked="false" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
<TextView android:layout_width="fill_parent"
    android:layout_height="150px"></TextView>
<AbsoluteLayout android:background="@drawable/footer_bar"
    android:layout_width="fill_parent" android:layout_height="wrap_content">
    <ImageButton android:id="@+id/make" android:background="@drawable/share"
        android:layout_x="270px" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <ImageButton android:id="@+id/cancel" android:layout_x="5px"
        android:background="@drawable/back" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
```

```
</AbsoluteLayout>
```

（2）编写程序文件 PlaySetting.java，其主要功能如下所示。

☑ 设置播放模式。

在设置播放模式时用的是 RadioGroup 组件，这个组件有单项选择的功能，里面有 RadioButton 项，多个 RadioButton 项只能同时选中一个，该播放器播放模式有单曲循环、随机播放、顺序播放等功能。MediaPlayer 有一个监听器，监听歌曲是否正在播放或者是否播放完成，当歌曲播放完成时，会触发方法 OnCompletionListener()，在该方法中可以处理歌曲播放完成后的操作。RadioGroup 可以进行单项选择操作。

☑ 歌词设置。

歌词是否显示是一个开关按钮 ToggleButton 实现的，有 ON 和 OFF 状态，当为 ON 时显示歌词，为 OFF 时关闭歌词显示。

ToogleButton 同样也有一个监听器，可以获得 ToogleButton 的不同状态。使用前对其进行实例化：(ToggleButton) View.findViewById(R.id.ly_lrc);，并且用“ToggleButton.isChecked();”获得开关状态。播放模式状态和歌词显示状态的操作结果都将以一个标志写在一个配置文件中，这是关于 Android 的存储方式。

了解文件 PlaySetting.java 的实现原理和功能后，其实现代码就非常容易理解了，主要代码如下所示。

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.setting);
        set_textView = (TextView) findViewById(R.id.setting);
        set_textView.setTextColor(Color.WHITE);
        sigle_Play = (RadioButton) findViewById(R.id.sigle_play);
        order_Play = (RadioButton) findViewById(R.id.order_play);
        random_Play = (RadioButton) findViewById(R.id.random_play);
        lyLrc = (ToggleButton) findViewById(R.id.ly_lrc);
        set_bt = (ImageButton) findViewById(R.id.make);
        cancel_bt = (ImageButton) findViewById(R.id.cancel);
        set_bt.setOnTouchListener(setting_bt_Listener);
        cancel_bt.setOnTouchListener(cancel_bt_Listener);
}

OnTouchListener setting_bt_Listener = new OnTouchListener() {//设置确定按钮监听器
        @Override
        public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                        v.setBackgroundResource(R.drawable.share_pressed);
                        SharedPreferences sp = getSharedPreferences("SET_MSG",
                                        MODE_WORLD_WRITEABLE);//文件共享
                        SharedPreferences.Editor editor = sp.edit();
                        if (sigle_Play.isChecked()) {
                                editor.putString("sigle_Play", "is_Sigle");
                                editor.putString("order_Play", null);
                                editor.putString("random_Play", null);
                        }
                        if (order_Play.isChecked()) {
                                editor.putString("sigle_Play", null);
                                editor.putString("order_Play", "is_Order");
                                editor.putString("random_Play", null);
                        }
                        if (random_Play.isChecked()) {
```

```
                    editor.putString("sigle_Play", null);
                    editor.putString("order_Play", null);
                    editor.putString("random_Play", "is_Random");
                }
                if (lyLrc.isChecked()) {
                    editor.putString("lyLrc", "is_Show");
                }
                if (!lyLrc.isChecked()) {
                    editor.putString("lyLrc", null);
                }
                editor.commit();//提交
                Intent intent = new Intent();
                setResult(4, intent);
            finish();
            } else if (event.getAction() == MotionEvent.ACTION_UP) {
                v.setBackgroundResource(R.drawable.share);
            }
            return false;
        }
};

OnTouchListener cancel_bt_Listener = new OnTouchListener() {//取消监听器
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            if (event.getAction() == MotionEvent.ACTION_DOWN) {
                v.setBackgroundResource(R.drawable.back_pressed);
            } else if (event.getAction() == MotionEvent.ACTION_UP) {
                v.setBackgroundResource(R.drawable.back);
                Intent intent = new Intent();
                setResult(3, intent);
            finish();
            }
            return false;
        }
};
```

6.4.6　设置显示歌词

显示歌词功能比较重要，所以笔者决定单独讲解其实现原理。本播放器的歌词是格式文件.lrc，查看.lrc 文件中的歌词格式如下所示。

```
[00:16.18]我爱你  心爱的姑娘
```

歌词格式是以“时间+歌词”的格式存储。

接下来将介绍如何将.lrc 中的歌词读取出来并存储在 Android 的配置文件中。

（1）用 XML 配置文件的存储

在 Android 系统中，SD 卡的目录结构如图 6-14 所示。

从图 6-14 中可以看到一个名为 sdcard 的目录，该目录即为扩展卡，里面预先存放着音频文件和.lrc 歌词文件，定义如下代码来指定.lrc 文件存在的路径，并将文件读取到 BufferReader 中。

```
BufferedReader buffer=new BufferedReader(new FileReader(new File("/sdcard/"+ musicName + ".lrc")));
```

由于要分开存放时间和歌词，所以应该定义两个 List<String>容器分别来存放时间和歌词。在读取歌词时，每次读取一行，再用算法将时间和歌词分开后放到一个数组里面，并分别存放在两个 list 中。由于歌曲在播放时会存在界面之间的跳转，所以歌词必须固定存放在一个文件中，而不能作为一个对象，因此，将两个时间 List 和歌词 List 再写进一个配置文件中。

Android 提供了共享文件类 SharedPreferences，它有一个方法 getSharedPreferences(参数 1，参数 2)，参数 1 表示写进时的名称标记，参数 2 表示读取模式，有只写模式（MODE_WORLD_WRITEABLE）和只读模式（MODE_WORLD_READABLE），在写之前将其设置为编辑状态，下面是使用静态方法设置编辑状态的代码。

```
SharedPreferences.Editor editor = sp.edit();
```

然后在对象 editor 中可以存入一个 HashMap<key,values>类型的键值，其格式是 putString(KEY, VALUES)，这样就可以将 List 中的对象转化成一样长的字符放进配置文件中。

当写入成功时，Android 系统会自动在目录 data/data/工程包名/shared_prefs/中生成一个配置文件。如图 6-15 所示。



图 6-14　SD 卡的目录结构　　　　图 6-15　配置文件

打开播放模式的 XML 配置文件，在此文件是以 MAP 的形式存储的，键名是<string name= "random_Play"></string>，其值是 is_Radom，如图 6-16 所示。

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
- <map>
    <string name="random_Play">is_Random</string>
    <string name="lyLrc">is_Show</string>
    <null name="order_Play" />
    <null name="sigle_Play" />
  </map>
```

图 6-16　XML 配置文件

（2）读取 XML 配置文件

仍以播放模式读取为例，当需要用到播放模式的确定时将读取.xml 文件，同样用共享文件类 SharedPreferences，通过用方法 getSharedPreferences("SET_MSG",MODE_WORLD_READABLE)并且是只读方式获得.xml 的文件内容。SharedPreferences 的对象调用方法是 getString("sigle_Play", null)，此方法会返回一个 String 类型的值，即以前存储进去的 String 值。当该标记不存在时，会默认返回一个 null 值。读取 XML 配置文件成功后，就可以根据配置的值对程序进行操作了。

6.4.7　文件浏览器模块

本项目程序实现了文件浏览器的功能，作为一个文件浏览器应该具有浏览功能。当程序运行到浏览界面时，会有各文件的目录显示及图标标识。从文件浏览器中能看到各文件，而且能对其进行操作，本程序

是专为播放器添加歌曲而设计的，故此功能仅限于对媒体文件的浏览和含有媒体文件的目录的浏览，所以功能比较局限。

当显示菜单界面时，通过新增选项进入到文件浏览器中，或者当播放列表为空时，会提示进入文件浏览器进行歌曲新增操作。其中文件浏览器界面如图 6-17 所示，SD 卡目录界面如图 6-18 所示。



图 6-17　文件浏览器界面

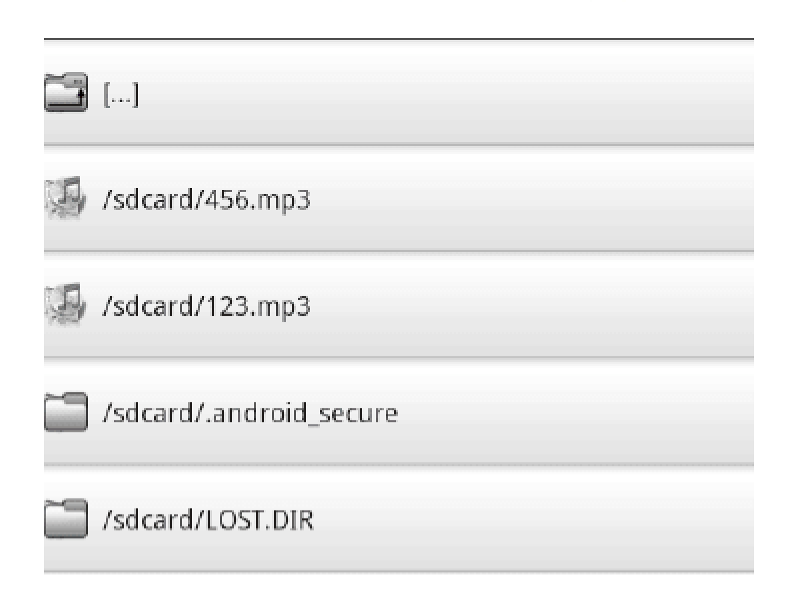


图 6-18　SD 卡目录界面

文件浏览器界面布局格式类似前面介绍的菜单，只是在界面的第一行新增了一个返回根目录的功能。由于程序只关系到目录/sdcard 下的文件，所以用程序屏蔽了其他的目录，这里只显示两个目录/sdcard 和/system。播放器只需要用到媒体文件，所以代码也屏蔽了其他文件的子目录。当选中 sdcard 会进入到图 6-18 所示界面，该目录下只显示媒体文件，如.mp3 和 SD 卡下的子目录。选中 system 会进入到图 6-17 所示界面，该目录会显示 system 下的各级子目录。当有媒体文件时才会出现添加 Dialog。

当要添加选中的歌曲时，程序有自动判断功能，首先弹出 Dialog。单击确定按钮后，程序会查询数据库中的歌曲，调用方法 query(fileName)，根据歌曲名字查询，如果歌曲不存在，则调用方法 insertMusic(file)，如果该歌曲名字已经存在，则弹出 Dialog 对话框。

（1）编写文件 directory_list.xml，实现文件浏览界面的布局，主要代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:background="@drawable/list_bg">
    <LinearLayout android:layout_width="fill_parent"
        android:gravity="center" android:layout_height="wrap_content"
        android:background="@drawable/footer_bar">
        <TextView android:text="SD 卡" android:id="@+id/store_card" android:textStyle="bold"
            android:layout_width="wrap_content" android:layout_height="wrap_content"
            android:textSize="@dimen/music_list_title"></TextView>
    </LinearLayout>

    <ListView android:id="@id/android:list" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
        <TextView android:layout_width="10px"
        android:layout_height="wrap_content" />
    <TextView android:id="@id/android:empty" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="@string/no_files"
/>
```

（2）编写文件 FileExplorerActivity.java，在此定义了文件浏览器类 FileExplorerActivity，此类继承了 ListActivity，此 Activity 是一个 ListView 界面。整个界面是一个 ListView 布局，而每一行是一个 LinearLayout 水平方式布局，上面将放置一个图片和一个文件全路径。该文件全路径被存放到数据库中，以便歌曲播放器能查询到歌曲路径源。

文件 FileExplorerActivity.java 的主要代码如下所示。

```
public class FileExplorerActivity extends ListActivity {
	private List<String> items = null;
	private TextView store_Card;
	@Override
	public void onCreate(Bundle icicle) {
		super.onCreate(icicle);
		setContentView(R.layout.directory_list);
		store_Card = (TextView) findViewById(R.id.store_card);
		store_Card.setTextColor(Color.WHITE);
		fill(new File("/").listFiles());
	}

	@Override
	protected void onListItemClick(ListView l, View v, int position, long id) {
		int selectionRowID = (int) position;
		File file = new File(items.get(selectionRowID));
		if (selectionRowID == 0) {
			fillWithRoot();
		} else {
			if (file.isDirectory()) {
				fill(file.listFiles());
			} else {
				Intent intent = this.getIntent();
				intent.putExtra("filePath", file);
				FileExplorerActivity.this.setResult(0, intent);
				showDialog("加入播放列表?", file);
			}
		}
	}

	private void fillWithRoot() {
		fill(new File("/").listFiles());
	}

	private void fill(File[ ] files) {
		items = new ArrayList<String>();
		items.add(getString(R.string.to_top));
		for (File file : files) {
			if (file.isDirectory()) {
				if ((file.getPath().indexOf("/sdcard")) != -1
						|| (file.getPath().indexOf("/system")) != -1)
					items.add(file.getPath());
			}
			if ((file.getPath().indexOf(".mp3")) != -1) {
				items.add(file.getPath());
			}
		}
		setListAdapter(new MusicAdapter(this, items));
	}
```

```
private void showDialog(String msg, final File file) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage(msg).setCancelable(false).setPositiveButton("确定",
                        new DialogInterface.OnClickListener() {
                                public void onClick(DialogInterface dialog, int id) {
                                        String fileName = file.getName().substring(0,
                                                        file.getName().indexOf("."));
                                        Log.i("info", fileName);
                                        if (query(fileName)) {
                                                insertMusic(file);//添加音乐
                                        }
                                }
                        }).setNegativeButton("取消", null);
        AlertDialog alert = builder.create();
        alert.show();
}

//添加音乐到播放列表
private final void insertMusic(File file) {
        ContentResolver cr = getContentResolver();
        ContentValues values = new ContentValues();
        Uri uri = DBProvider.CONTENT_URI;
        String fileName = file.getName().substring(0,
                        file.getName().indexOf("."));
        values.put(FileColumn.NAME, fileName);
        values.put(FileColumn.PATH, file.getAbsolutePath());
        values.put(FileColumn.TYPE, "Music");
        values.put(FileColumn.SORT, "popular");
        cr.insert(uri, values);
        Toast.makeText(FileExplorerActivity.this, "已加入", Toast.LENGTH_LONG)
                        .show();
        Intent intent = new Intent();
setResult(6, intent);
finish();
}

private boolean query(String name) {
        ContentResolver cr = getContentResolver();
        Uri uri = DBProvider.CONTENT_URI;
        String[ ] projection = { "filename" };
        Cursor c = cr.query(uri, projection, null, null, null);
        if (c.moveToFirst()) {
                for (int i = 0; i < c.getCount(); i++) {
                        c.moveToPosition(i);
                        String filename_DB = c.getString(0);
                        if (name.equals(filename_DB)) {//判断播放列表中是否存在该歌曲
                                AlertDialog.Builder builder = new AlertDialog.Builder(this);
                                builder.setMessage("文件已存在").setCancelable(false)
                                                .setPositiveButton("返回列表",
                                                                new DialogInterface.OnClickListener() {
                                                                        public void onClick(
```

```
                                                        DialogInterface dialog, int id) {
                                                            Intent intent = new Intent(
                                                            FileExplorerActivity.this,
                                                            PlayListActivity.class);
                                                    startActivity(intent);
                                                }
                                            }).setNegativeButton("重新添加", null);
                        AlertDialog alert = builder.create();
                        alert.show();
                        return false;
                    }
                }
            }
            return true;
        }
```

上述 ListView 实现了自动判断的功能，即程序可以通过访问扩展卡中的文件属性而自动识别文件属性。当为一个 MP3 格式文件时，则前面图标显示 MP3 图标；当为一个文件目录时，则图标标识为一个文件。文件浏览器是用递归算法实现的，其中 fillWithRoot()用于返回根目录的列表。

6.4.8 数据存储

在播放器正常运行时，由于各界面存在相互跳转，为了避免数据在界面跳转的过程中丢失，需要将一些数据进行临时存储或者永久存储。

Android 作为一种手机操作系统，提供了 Preference（配置）、File（文件）、SQLite 数据和网络等存取数据的方式。

另外，在 Android 中各个应用程序组件之间是相互独立的，彼此的数据不能共享。为了实现数据的共享，Android 提供了 Content Provider 组件来实现应用程序之间数据的共享。

（1）SharedPreferences 存储方式

SharedPreference 提供了一种轻量级的数据存取方法，一般数据比较少，仅保存一些简单的配置信息。以“键-值”（是一个 Map）对的方式，将数据保存在一个 XML 配置文件中。

在本实例中用到了如下两种数据存储接口。

☑ android.content.SharedPreferences：提供了保存数据的方法。

☑ android.content.SharedPreferences.Editor：提供了获得数据的方法。

注意： 上述有关SharedPreferences数据存储的知识，请读者参阅相关资料，这并不是本书的重点。

以播放器中的播放模式存取为例，实现流程如下所示。

☑ XML 配置文件的读取。

仍以播放模式读取为例，当需要用到播放模式的确定时，将读取.xml 文件，同样用共享文件类 SharedPreferences，通过用方法 getSharedPreferences("SET_MSG",MODE_WORLD_READABLE)，并且是只读方式获得 XML 文件的内容。SharedPreferences 的对象调用方法 getString("sigle_Play", null)，该方法返回一个 String 类型的值，即以前存储的 String 值。当该标记不存在时，此方法会默认返回一个 null 值。获得成功后就可以运用当前的值再对程序进行操作了。

☑ XML 配置文件的存储。

在类 SharedPreferences 中有一个方法 getSharedPreferences(参数 1，参数 2)，参数 1 为写进时的标记，



参数 2 为读取模式，有只写模式（MODE_WORLD_WRITEABLE）和只读模式（MODE_WORLD_READABLE），在写之前将其置入编辑状态，使用静态方法：SharedPreferences.Editor editor = sp.edit();对象 editor 可以存入一个 HashMap<key,values>类型的键值，即 putString(KEY, VALUES)，这样，可以将 List 中的对象转化成一样长的字符中放进配置文件中。当写入成功时，Android 系统会自动在目录“data/data/工程包名/shared_prefs”下生成一个配置文件。

（2）File 存储方式

File 存储方式可以将一些数据直接以文件的形式保存在设备中，例如，一些文本文件、PDF 文件、音视频文件和图片等。Android 提供了如下文件读写的方法。

☑ Context.openFileInput()：获得标准 Java 文件输入流（FileInputStream）。

☑ Context.openFileOutput()：获得标准 Java 文件输出流（FileOutputStream）。

☑ Resources.openRawResource (R.raw.myDataFile)：返回 InputStream。

（3）SQLiteDatabase 数据库

SQLite 是一个嵌入式数据库引擎，针对内存等资源有限的设备（如手机、PDA、MP3）提供的一种高效的数据库引擎。SQLite 数据库不像其他的数据库（例如 Oracle），它没有服务器进程，所有的内容包含在同一个单文件中。该文件是跨平台的，可以自由复制。基于其自身的先天优势，SQLite 在嵌入式领域得到了广泛应用。

☑ SQLiteDatabase 类：代表一个数据库对象，在里面提供了操作数据库的一些方法，具体方法请读者参考相关教材或书籍。

☑ SQLiteOpenHelper 类：是 SQLiteDatabase 的一个帮助类，用来管理数据库的创建和版本更新。一般的用法是定义一个类继承，并实现其两个抽象方法 onCreate(SQLiteDatabase db)和 onUpgrade (SQLiteDatabase db, int oldVersion, int newVersion)来创建和更新数据库。

Android 的 3 种数据存储方式则让用户可以轻松方便地进行程序编写和数据访问，更不会使数据丢失，这对程序书写有很大帮助。

第 7 章　魔 塔 游 戏

益智游戏是指通过一定的逻辑或是数学、物理、化学，甚至是自己设定的原理来完成一定任务的小游戏。一般会比较有意思，需要适当的思考，适合年轻人玩。益智游戏通常以游戏的形式锻炼了游戏者的脑、眼、手等，使其增强自身的逻辑分析能力和思维敏捷性。值得一提的是，优秀的益智游戏娱乐性也十分强，既好玩又耐玩。在本章的内容中，将详细讲解益智类游戏——魔塔游戏的具体实现过程。

7.1　魔 塔 简 介

知识点讲解：光盘:视频\视频讲解\第 7 章\魔塔简介.avi

魔塔是一种固定数值 RPG 游戏，这个游戏需要动很多脑筋，任何一个轻率的选择都可能导致游戏的失败，还可以锻炼游戏者的数学能力。在本节的内容中，将简要介绍此款游戏的基本知识。

7.1.1　游戏简介

魔塔是一类固定数值的 RPG 游戏。经典魔塔有 5 种，分别是 Tower of the Sorcerer、24 层魔塔（胖老鼠制作）、新新魔塔（cos180 制作）、魔塔 2006（Oksh 制作）、魔塔 2000（ANGELA 制作）。虽然魔塔的界面很像是一般的地牢游戏，貌似可以轻松过关，但事实上玩这个游戏需要动很多脑筋，任何一个轻率的选择都可能导致游戏的失败，该游戏属性有攻击、防御、生命、金币、经验。对怪物的伤害次数计算公式，即敌人的生命/(自己的攻击-敌人的防御)；而伤害的数目计算是怪物伤害次数*(敌人的攻击-自己的防御)。这些怪物的伤害除了新新魔塔，其他的是固定的，所以魔塔可以算作是益智类中最好的游戏。商店一般有 3 种，21 层魔塔里商店中物品是固定的价格，50 层魔塔里商店中物品是价钱翻倍，新新魔塔里商店有两个，一个初始 20 金币，买一次贵 1 金币；一个初始 50 金币，买一次贵 2 金币。魔塔游戏虽不大，但是制作精美，道具很多，而且难度不低，考验智商。

7.1.2　发展版本

魔塔游戏最早是由两位日本设计师制作的，引进中国后胖老鼠工作室又制作了 21 层魔塔，然后陆续有新型作品新新魔塔、魔塔 2000，后来 Oksh 制作了 RMXP 用的魔塔样板，参与制作魔塔的人也就越来越多了，但制作水平不一。

（1）21 层魔塔和 24 层魔塔

这两款魔塔都是由胖老鼠工作室制作的，其区别从名字中就可以看出来，即层数不一样。但是如果不在 25 分钟之内打到 16 层并获得灵杖，或在获得灵杖之前已经拿到十字架，就不能出现隐藏楼层。其他方面并没有太大区别。在 24 层魔塔中，圣光盾的价格被降低为 500 金币，不同于 21 层魔塔的 800 金币，从此打法就发生了变化（节省了 3 金币）。21 层冥灵魔王的能力被改为 2550/2250。另外还有一个不太容易看出来的区别：多了一种宝物，目前不知道叫什么，用 C 键使用，可以破坏整个楼层的墙，现在还没有人发

现获得此宝物的方法。除此之外，还可以在游戏中存档，不怕打错一步重来了。

（2）21 层魔塔和扑克魔塔一

这两款魔塔游戏基本上是一样的，地图几乎完全相同，但是刘景雄在制作扑克魔塔一时做了如下修改。

☑ 4 层的两个骷髅人下面各多了一个红血瓶。

☑ 11 层守卫黄金盾的士兵改成了 4 个中级卫兵。

☑ 困难模式下，21 层的两个栅栏门被开成了两个绿花门，在心形的两边也多出了两个 NPC。

（3）魔塔在中国

2003 年胖老鼠工作室发布了第一款中文版魔塔，该魔塔除“序章”外共 21 层，依旧以勇士救公主为背景，最终能够彻底打败魔王救出公主。

2004 年胖老鼠工作室又发布了该游戏的 1.1 版和 1.2 版，把魔塔层数增加到 24 层。游戏使用方向键操作，空格和小键盘 5 为确定键，小键盘 8 和 2 做上下选择。

7.2　设计游戏框架

知识点讲解：光盘:视频\视频讲解\第 7 章\设计游戏框架.avi

从本节内容开始，将详细讲解在 Android 平台上开发一个魔塔游戏的具体流程。因为所有游戏是基于框架的，所以设计一个合适的框架尤为重要。为了正确设计框架，先看市面上魔塔游戏的界面，如图 7-1 所示。

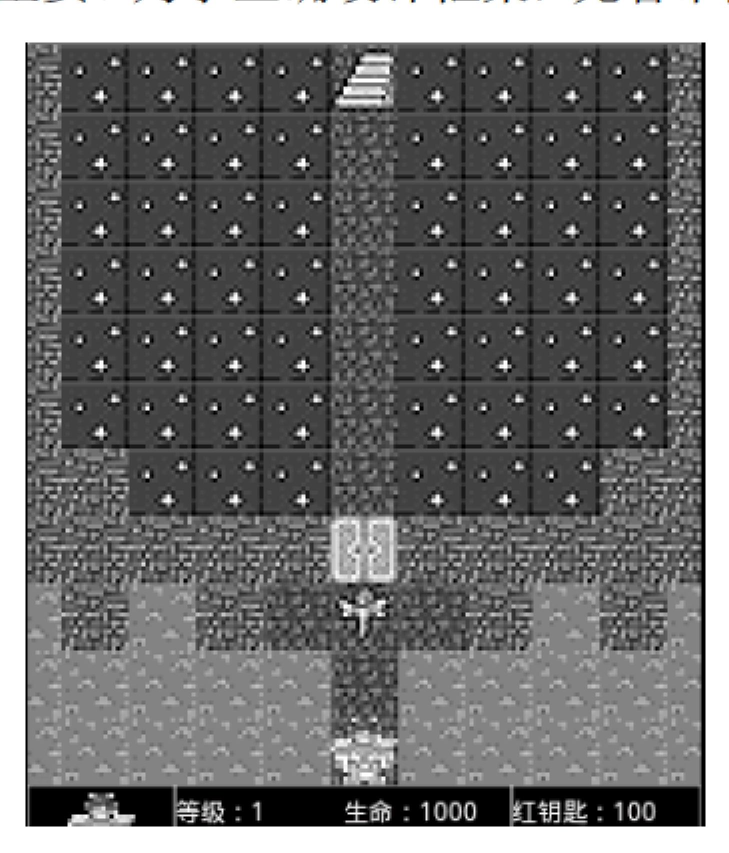


图 7-1　游戏界面

由游戏界面可知，游戏中包含了地图、角色、屏幕界面、道具等元素，上述元素构成了一个视图，例如屏幕视图、道具视图、角色视图等。

7.2.1　设计界面视图

在 Android 中，视图是通过继承 View 类实现的，在 View 类中包含了各种绘制图形的方法和事件处理，这样构建一个游戏界面类将变得轻而易举。界面类 ViewCH 的具体代码如下所示。

```
public abstract class ViewCH extends View
{
	public GameView(Context context)
	{
```

```
        super(context);
    }
    /*绘图*/
    protected abstract void onDraw(Canvas canvas);
    /*按键按下*/
    public abstract boolean onKeyDown(int keyCode);
    /*按键弹起*/
    public abstract boolean onKeyUp(int keyCode);
    /*回收资源*/
    protected abstract void reCycle();
    /*刷新*/
    protected abstract void refurbish();
}
```

7.2.2 屏幕处理

在玩游戏的过程中，屏幕会随玩家的操作而变化。在设计好整个游戏的屏幕框架后，还需要设计屏幕的控制类。通过这些类，可以根据不同的游戏状态来控制屏幕的具体显示内容。在此编写 MainCH 类，实现文件是 MainCH.java，具体代码如下所示。

```
public class MainCH
{
    private static ViewCH m_GameView = null;            //当前需要显示的对象
    private Context m_Context = null;
    private MagicCH m_MagicTower = null;
    private int m_status = -1;                          //游戏状态
    public   PlayerCH mPlayerCH;
    public byte mbMusic = 0;
    public MainCH(Context context)
    {
        m_Context = context;
        m_MagicTower = (MagicCH)context;
        m_status = -1;
        initGame();
    }
    //初始化游戏
    public void initGame()
    {
        controlView(yaCH.GAME_SPLASH);
        mPlayerCH = new PlayerCH(m_MagicTower);
    }
    //得到游戏状态
    public int getStatus()
    {
        return m_status;
    }
    //设置游戏状态
    public void setStatus(int status)
    {
        m_status = status;
    }
```

```
//得到主类对象
public Activity getMagicTower()
{
    return m_MagicTower;
}
//得到当前需要显示的对象
public static ViewCH getMainView()
{
   return m_GameView;
}
//控制显示什么界面
public void controlView(int status)
{
    if(m_status != status)
    {
        if(m_GameView != null)
        {
            m_GameView.reCycle();
            System.gc();
        }
    }
    freeGameView(m_GameView);
    switch (status)
    {
    case yaCH.GAME_SPLASH:
        m_GameView = new SplashCH(m_Context,this);
        break;
    case yaCH.GAME_MENU:
        m_GameView = new MenuCH(m_Context,this);
        break;
    case yaCH.GAME_HELP:
        m_GameView = new HelpScreen(m_Context,this);
        break;
    case yaCH.GAME_ABOUT:
        m_GameView = new AboutCH(m_Context,this);
        break;
    case yaCH.GAME_RUN:
        m_GameView = new ScreenCH(m_Context,m_MagicTower,this,true);
        break;
    case yaCH.GAME_CONTINUE:
        m_GameView = new ScreenCH(m_Context,m_MagicTower,this,false);
        break;
    }
    setStatus(status);
}
//释放界面对象
public void freeGameView(ViewCH gameView)
{
    if(gameView != null)
    {
        gameView = null;
```

```
                System.gc();
            }
        }
}
```

7.2.3 更新线程

到此为止，整个游戏界面的设计告一段落。但是还有一个十分关键的问题，屏幕更新功能需要借助线程来实现，因为只有这样才能在本质上实现实时更新。在本游戏项目中开启了一个主线程，并通过方法getMainView()来获取并显示当前界面，而且可以根据不同的界面来进行界面更新。线程更新功能是在文件CanvasCH.java 中实现的，具体代码如下所示。

```
public class CanvasCH extends View implements Runnable
{
        private String    m_Tag = "ThreadCanvas_Tag";
        public CanvasCH(Context context)
        {
                super(context);
        }
        /*绘图*/
        protected void onDraw(Canvas canvas)
        {
                if (MainCH.getMainView() != null)
                {
                        MainCH.getMainView().onDraw(canvas);
                }
                else
                {
                        Log.i(m_Tag, "null");
                }
        }


        /*绘图显示*/
        public void start()
        {
                Thread t = new Thread(this);
                t.start();
        }
        //刷新界面
        public void refurbish()
        {
                if (MainCH.getMainView() != null)
                {
                        MainCH.getMainView().refurbish();
                }
        }


        /*游戏循环*/
```

```
        public void run()
        {
                while (true)
                {
                        try
                        {
                                Thread.sleep(yaCH.GAME_LOOP);
                        }
                        catch (Exception e)
                        {
                                e.printStackTrace();
                        }
                        refurbish(); //更新显示
                        postInvalidate(); //刷新屏幕
                }
        }
        //按键处理（按键按下）
        boolean onKeyDown(int keyCode)
        {
                if (MainCH.getMainView() != null)
                {
                        MainCH.getMainView().onKeyDown(keyCode);
                }
                else
                {
                        Log.i(m_Tag, "null");
                }
                return true;
        }
        //按键弹起
        boolean onKeyUp(int keyCode)
        {
                if (MainCH.getMainView() != null)
                {
                        MainCH.getMainView().onKeyUp(keyCode);
                }
                else
                {
                        Log.i(m_Tag, "null");
                }
                return true;
        }
}
```

7.2.4　游戏界面显示

经过前面的界面类设计工作后，要想完美实现游戏界面，还需要做最后一步工作。接下来需要用一个 Activity 来显示具体的界面。因为是在 CanvasCH 中控制界面的，所以只需使用方法 setContentView()来显示一个 CanvasCH 对象即可。游戏界面显示功能通过文件 MagicCH.java 实现的，具体代码如下所示。

```
public class MagicCH extends Activity
```

```
{
	private CanvasCH mThreadCanvas = null;
	public void onCreate(Bundle savedInstanceState)
	{
		super.onCreate(savedInstanceState);
		setTheme(android.R.style.Theme_Black_NoTitleBar_Fullscreen);
		requestWindowFeature(Window.FEATURE_NO_TITLE);
		getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.Layout
Params.FLAG_FULLSCREEN);
		new MainCH(this);
		mThreadCanvas = new CanvasCH(this);
		setContentView(mThreadCanvas);
	}

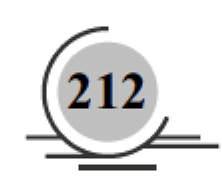
	/*暂停*/
	protected void onPause()
	{
		super.onPause();
	}
	/*重绘*/
	protected void onResume()
	{
		super.onResume();
		mThreadCanvas.requestFocus();
		mThreadCanvas.start();
	}
	/*按键按下*/
	public boolean onKeyDown(int keyCode, KeyEvent event)
	{
		mThreadCanvas.onKeyDown(keyCode);
		return false;
	}
	/*按键弹起*/
	public boolean onKeyUp(int keyCode, KeyEvent event)
	{
		mThreadCanvas.onKeyUp(keyCode);
		return false;
	}
}
```

通过上述处理，一个基本的游戏框架建设完毕。在后续的开发中，只需直接继承上面的类界面即可。即继承 GameView，然后在 MainView 中更改游戏状态即可。

7.3 绘 制 处 理

知识点讲解：光盘:视频\视频讲解\第 7 章\绘制处理.avi

设计好游戏框架之后，需要绘制游戏中的角色和场景。在本节的内容中，将简要介绍魔塔游戏中绘制处理的实现过程。



7.3.1　绘制地图

绘制地图功能是通过文件 TiledCH.java 实现的，在其中定义了方法 paint()来绘制地图，并将绘制的图形显示在屏幕上。文件 TiledCH.java 的具体实现流程如下所示。

（1）定义方法 TiledCH()，其中参数 columns 和 rows 分别表示地图的列数和行数，即构建的地图数组的行数和列数；参数 image 表示地图所有图块的一张图片，即在编辑地图时所使用的图片；参数 tileWidth 和 tileHeight 分别表示每个图片的宽度和高度。对应代码如下所示。

```
public TiledCH(int columns, int rows, Bitmap image, int tileWidth,
			int tileHeight) {
		super(columns < 1 || tileWidth < 1 ? -1 : columns * tileWidth, rows < 1
				|| tileHeight < 1 ? -1 : rows * tileHeight);
		if (((image.getWidth() % tileWidth) != 0)
				|| ((image.getHeight() % tileHeight) != 0)) {
			throw new IllegalArgumentException();
		}
		this.columns = columns;
		this.rows = rows;
		cellMatrix = new int[rows][columns];
		int noOfFrames = (image.getWidth() / tileWidth)
				* (image.getHeight() / tileHeight);
		createStaticSet(image, noOfFrames + 1, tileWidth, tileHeight, true);
}
```

（2）定义方法 setCell()来设置该地图使用的地图数据，通过一个循环设置了所有要显示的行和列的图块索引值。参数 col 和 row 分别表示在屏幕上要显示的列数和行数；参数 tileIndex 则表示在地图上显示图块的值。对应代码如下所示。

```
public void setCell(int col, int row, int tileIndex) {
		if (col < 0 || col >= this.columns || row < 0 || row >= this.rows) {
			throw new IndexOutOfBoundsException();
		}
		if (tileIndex > 0) {
			if (tileIndex >= numberOfTiles) {
				throw new IndexOutOfBoundsException();
			}
		} else if (tileIndex < 0) {
			if (anim_to_static == null || (-tileIndex) >= numOfAnimTiles) {
				throw new IndexOutOfBoundsException();
			}
		}
		cellMatrix[row][col] = tileIndex;
}
```

（3）定义方法 fillCells()在图层中指定单元格区域，对应部分代码如下所示。

```
public void fillCells(int col, int row, int numCols, int numRows,
			int tileIndex) {
		if (col < 0 || col >= this.columns || row < 0 || row >= this.rows
				|| numCols < 0 || col + numCols > this.columns || numRows < 0
				|| row + numRows > this.rows) {
			throw new IndexOutOfBoundsException();
		}
```

```
        if (tileIndex > 0) {
            if (tileIndex >= numberOfTiles) {
                throw new IndexOutOfBoundsException();
            }
        } else if (tileIndex < 0) {
            if (anim_to_static == null || (-tileIndex) >= numOfAnimTiles) {
                throw new IndexOutOfBoundsException();
            }
        }
        for (int rowCount = row; rowCount < row + numRows; rowCount++) {
            for (int columnCount = col; columnCount < col + numCols; columnCount++) {
                cellMatrix[rowCount][columnCount] = tileIndex;
            }
        }
```

（4）定义方法 paint()和 drawImage()将地图绘制到屏幕上。此功能通过一个循环将设置的地图数组和图片进行对应，并绘制到对应的屏幕上。对应代码如下所示。

```
    public final void paint(Canvas canvas) {
        if (canvas == null) {
            throw new NullPointerException();
        }
        if (visible) {
            int tileIndex = 0;
            int ty = this.y;
            for (int row = 0; row < cellMatrix.length; row++, ty += cellHeight) {
                int tx = this.x;
                int totalCols = cellMatrix[row].length;
                for (int column = 0; column < totalCols; column++, tx += cellWidth) {
                    tileIndex = cellMatrix[row][column];
                    if (tileIndex == 0) { // transparent tile
                        continue;
                    } else if (tileIndex < 0) {
                        tileIndex = getAnimatedTile(tileIndex);
                    }
                    drawImage(canvas, tx, ty, sourceImage, tileSetX[tileIndex],
                            tileSetY[tileIndex], cellWidth, cellHeight);
                }
            }
        }
    }
    private void drawImage(Canvas canvas, int x, int y,
            Bitmap bsrc, int sx, int sy, int w, int h) {
        Rect rect_src = new Rect();
        rect_src.left = sx;
        rect_src.right = sx + w;
        rect_src.top = sy;
        rect_src.bottom = sy + h;
        Rect rect_dst = new Rect();
        rect_dst.left = x;
        rect_dst.right = x + w;
        rect_dst.top = y;
```

```
            rect_dst.bottom = y + h;
            canvas.drawBitmap(bsrc, rect_src, rect_dst, null);
            rect_src = null;
            rect_dst = null;
    }
```

7.3.2　绘制游戏主角

魔塔游戏中的主角被称为“精灵”，这个主角可以做很多动作，例如，可以在向 4 个方向移动，并且分别对应 4 个不同的动画。动画本身就是将图片一帧一帧地连接起来，然后循环地播放每一帧。在本实例中，是通过 SpriteCH 类实现主角功能的。SpriteCH 类是一个显示图像类，该类和 TiledCH 的区别是：SpriteCH 是由一个可以有好几帧的图组成的，所以 SpriteCH 通常被定义一些小的、有动作的游戏对象，而 TiledCH 通常被用来构造生动的背景。SpriteCH 类是在文件 SpriteCH.java 中实现的，此文件的具体实现流程如下所示。

（1）定义系统所需要的初始常量和标量，具体代码如下所示。

```
public static final int TRANS_NONE = 0;
public static final int TRANS_ROT90 = 5;
public static final int TRANS_ROT180 = 3;
public static final int TRANS_ROT270 = 6;
public static final int TRANS_MIRROR = 2;
public static final int TRANS_MIRROR_ROT90 = 7;
public static final int TRANS_MIRROR_ROT180 = 1;
public static final int TRANS_MIRROR_ROT270 = 4;
private static final int INVERTED_AXES = 0x4;
private static final int X_FLIP = 0x2;
private static final int Y_FLIP = 0x1;
private static final int ALPHA_BITMASK = 0xff000000;
Bitmap sourceImage;
int numberFrames;
int[] frameCoordsX;
int[] frameCoordsY;
int srcFrameWidth;
int srcFrameHeight;
int[ ] frameSequence;
private int sequenceIndex; //= 0
private boolean customSequenceDefined; //= false;
int dRefX; //=0
int dRefY; //=0
int collisionRectX; //=0
int collisionRectY; //=0
int collisionRectWidth;
int collisionRectHeight;
int t_currentTransformation;
int t_collisionRectX;
int t_collisionRectY;
int t_collisionRectWidth;
int t_collisionRectHeight;
```

（2）在 SpriteCH 类中构造如下 3 个构造方法，具体代码如下所示。

```
public SpriteCH(Bitmap image) {
        super(image.getWidth(), image.getHeight());
        initializeFrames(image, image.getWidth(), image.getHeight(), false);
        this.setTransformImpl(TRANS_NONE);
}

public SpriteCH(Bitmap image, int frameWidth, int frameHeight) {
        super(frameWidth, frameHeight);
        if ((frameWidth < 1 || frameHeight < 1)
                        || ((image.getWidth() % frameWidth) != 0)
                        || ((image.getHeight() % frameHeight) != 0)) {
                throw new IllegalArgumentException();
        }
        initializeFrames(image, frameWidth, frameHeight, false);
        initCollisionRectBounds();
        this.setTransformImpl(TRANS_NONE);
}

public SpriteCH(SpriteCH s) {
        super(s != null ? s.getWidth() : 0, s != null ? s.getHeight() : 0);
        if (s == null) {
                throw new NullPointerException();
        }
        this.sourceImage = s.sourceImage;
        this.numberFrames = s.numberFrames;
        this.frameCoordsX = new int[this.numberFrames];
        this.frameCoordsY = new int[this.numberFrames];
        System.arraycopy(s.frameCoordsX, 0, this.frameCoordsX, 0, s
                        .getRawFrameCount());
        System.arraycopy(s.frameCoordsY, 0, this.frameCoordsY, 0, s
                        .getRawFrameCount());
        this.x = s.getX();
        this.y = s.getY();
        this.dRefX = s.dRefX;
        this.dRefY = s.dRefY;
        this.collisionRectX = s.collisionRectX;
        this.collisionRectY = s.collisionRectY;
        this.collisionRectWidth = s.collisionRectWidth;
        this.collisionRectHeight = s.collisionRectHeight;
        this.srcFrameWidth = s.srcFrameWidth;
        this.srcFrameHeight = s.srcFrameHeight;
        setTransformImpl(s.t_currentTransformation);
        this.setVisible(s.isVisible());
        this.frameSequence = new int[s.getFrameSequenceLength()];
        this.setFrameSequence(s.frameSequence);
        this.setFrame(s.getFrame());
        this.setRefPixelPosition(s.getRefPixelX(), s.getRefPixelY());
}
```

在上述 3 个构造方法中，参数 image 表示主角的图片，参数 frameWidth 和 frameHeight 分别表示主角图片的每一帧的宽度和高度，参数 s 表示通过一个主角来创建另一个主角。在构造 SpriteCH 类时，需要指定

主角的高度和宽度，单位是像素。图像的高度和宽度必须分别是主角的高度和宽度的整数倍。也就是说，需要正好让计算机把图像按照主角的大小划分成几个类。本游戏中的帧无论是横排、竖排、横竖排、方阵排都无影响。接下来就可以指定帧数了，左上方是编号 0，然后从左到右、从上到下依次排列。在项目中使用方法 setFrame(int sequenceIndex)来设置哪一帧被显示，此功能是通过将编号作为参数传递实现的。

（3）因为使用类 TiledCH 可自动根据主角的位置来判断地图绘制的位置，所以可以使用 setRefPixelPosition()方法来设置主角的位置。其中，参数 x 和 y 是主角的位置。

```
public void setRefPixelPosition(int x, int y) {
        this.x = x
                - getTransformedPtX(dRefX, dRefY, this.t_currentTransformation);
        this.y = y
                - getTransformedPtY(dRefX, dRefY, this.t_currentTransformation);
}
```

（4）定义如下 3 个碰撞检测函数，分别表示主角和 TiledCH 的碰撞、主角和主角的碰撞、主角和图片的碰撞。参数 pixelLevel 表示使用像素检测还是矩形检测。矩形检测只需要将主角对应成相应的矩形范围来进行检查，这种检测速度很快，但不是很准确，对于碰撞要求不高的游戏可以使用。同时还可以将一个 SpriteCH 分解成很多矩形来使用矩形检测以提高准确性。而像素检测则比较准确，但是速度必然会减慢。部分实现代码如下所示。

```
public final boolean collidesWith(SpriteCH s, boolean pixelLevel) {
        if (!(s.visible && this.visible)) {
                return false;
        }
        int otherLeft = s.x + s.t_collisionRectX;
        int otherTop = s.y + s.t_collisionRectY;
        int otherRight = otherLeft + s.t_collisionRectWidth;
        int otherBottom = otherTop + s.t_collisionRectHeight;
        int left = this.x + this.t_collisionRectX;
        int top = this.y + this.t_collisionRectY;
        int right = left + this.t_collisionRectWidth;
        int bottom = top + this.t_collisionRectHeight;
        if (intersectRect(otherLeft, otherTop, otherRight, otherBottom, left,
                        top, right, bottom)) {
                if (pixelLevel) {
                        if (this.t_collisionRectX < 0) {
                                left = this.x;
                        }
                        if (this.t_collisionRectY < 0) {
                                top = this.y;
                        }
                        if ((this.t_collisionRectX + this.t_collisionRectWidth) > this.width) {
                                right = this.x + this.width;
                        }
                        if ((this.t_collisionRectY + this.t_collisionRectHeight) > this.height) {
                                bottom = this.y + this.height;
                        }
                        if (s.t_collisionRectX < 0) {
                                otherLeft = s.x;
                        }
                        if (s.t_collisionRectY < 0) {
```

```
                        otherTop = s.y;
                    }
                    if ((s.t_collisionRectX + s.t_collisionRectWidth) > s.width) {
                        otherRight = s.x + s.width;
                    }
                    if ((s.t_collisionRectY + s.t_collisionRectHeight) > s.height) {
                        otherBottom = s.y + s.height;
                    }
                    if (!intersectRect(otherLeft, otherTop, otherRight,
                            otherBottom, left, top, right, bottom)) {
                        return false;
                    }
                    int intersectLeft = (left < otherLeft) ? otherLeft : left;
                    int intersectTop = (top < otherTop) ? otherTop : top;
                    int intersectRight = (right < otherRight) ? right : otherRight;
                    int intersectBottom = (bottom < otherBottom) ? bottom: otherBottom;
                    int intersectWidth = Math.abs(intersectRight - intersectLeft);
                    int intersectHeight = Math.abs(intersectBottom - intersectTop);
                    int thisImageXOffset = getImageTopLeftX(intersectLeft,
                            intersectTop, intersectRight, intersectBottom);
                    int thisImageYOffset = getImageTopLeftY(intersectLeft,
                            intersectTop, intersectRight, intersectBottom);
                    int otherImageXOffset = s.getImageTopLeftX(intersectLeft,
                            intersectTop, intersectRight, intersectBottom);
                    int otherImageYOffset = s.getImageTopLeftY(intersectLeft,
                            intersectTop, intersectRight, intersectBottom);
                    return doPixelCollision(thisImageXOffset, thisImageYOffset,
                            otherImageXOffset, otherImageYOffset, this.sourceImage,
                            this.t_currentTransformation, s.sourceImage,
                            s.t_currentTransformation, intersectWidth,
                            intersectHeight);
                } else {
                    return true;
                }
            }
            return false;
        }

        public final boolean collidesWith(TiledCH t, boolean pixelLevel) {
            if (!(t.visible && this.visible)) {
                return false;
            }
            int tLx1 = t.x;
            int tLy1 = t.y;
            int tLx2 = tLx1 + t.width;
            int tLy2 = tLy1 + t.height;
            int tW = t.getCellWidth();
            int tH = t.getCellHeight();
            int sx1 = this.x + this.t_collisionRectX;
            int sy1 = this.y + this.t_collisionRectY;
            int sx2 = sx1 + this.t_collisionRectWidth;
```

```
int sy2 = sy1 + this.t_collisionRectHeight;
int tNumCols = t.getColumns();
int tNumRows = t.getRows();
int startCol;
int endCol;
int startRow;
int endRow;
if (!intersectRect(tLx1, tLy1, tLx2, tLy2, sx1, sy1, sx2, sy2)) {
    return false;
}

startCol = (sx1 <= tLx1) ? 0 : (sx1 - tLx1) / tW;
startRow = (sy1 <= tLy1) ? 0 : (sy1 - tLy1) / tH;
endCol = (sx2 < tLx2) ? ((sx2 - 1 - tLx1) / tW) : tNumCols - 1;
endRow = (sy2 < tLy2) ? ((sy2 - 1 - tLy1) / tH) : tNumRows - 1;
if (!pixelLevel) {
    // check for intersection with a non-empty cell
    for (int row = startRow; row <= endRow; row++) {
        for (int col = startCol; col <= endCol; col++) {
            if (t.getCell(col, row) != 0) {
                return true;
            }
        }
    }
    return false;
} else {
    if (this.t_collisionRectX < 0) {
        sx1 = this.x;
    }
    if (this.t_collisionRectY < 0) {
        sy1 = this.y;
    }
    if ((this.t_collisionRectX + this.t_collisionRectWidth) > this.width) {
        sx2 = this.x + this.width;
    }
    if ((this.t_collisionRectY + this.t_collisionRectHeight) > this.height) {
        sy2 = this.y + this.height;
    }
    if (!intersectRect(tLx1, tLy1, tLx2, tLy2, sx1, sy1, sx2, sy2)) {
        return (false);
    }
    startCol = (sx1 <= tLx1) ? 0 : (sx1 - tLx1) / tW;
    startRow = (sy1 <= tLy1) ? 0 : (sy1 - tLy1) / tH;

    endCol = (sx2 < tLx2) ? ((sx2 - 1 - tLx1) / tW) : tNumCols - 1;
    endRow = (sy2 < tLy2) ? ((sy2 - 1 - tLy1) / tH) : tNumRows - 1;
    int cellTop = startRow * tH + tLy1;
    int cellBottom = cellTop + tH;
    int tileIndex;
    for (int row = startRow; row <= endRow; row++, cellTop += tH, cellBottom += tH) {
        int cellLeft = startCol * tW + tLx1;
```

```
                int cellRight = cellLeft + tW;
                for (int col = startCol; col <= endCol; col++, cellLeft += tW, cellRight += tW) {
                    tileIndex = t.getCell(col, row);
                    if (tileIndex != 0) {
                        int intersectLeft = (sx1 < cellLeft) ? cellLeft : sx1;
                        int intersectTop = (sy1 < cellTop) ? cellTop : sy1;
                        int intersectRight = (sx2 < cellRight) ? sx2
                                : cellRight;
                        int intersectBottom = (sy2 < cellBottom) ? sy2
                                : cellBottom;
                        if (intersectLeft > intersectRight) {
                            int temp = intersectRight;
                            intersectRight = intersectLeft;
                            intersectLeft = temp;
                        }
                        if (intersectTop > intersectBottom) {
                            int temp = intersectBottom;
                            intersectBottom = intersectTop;
                            intersectTop = temp;
                        }
                        int intersectWidth = intersectRight - intersectLeft;
                        int intersectHeight = intersectBottom - intersectTop;
                        int image1XOffset = getImageTopLeftX(intersectLeft,
                                intersectTop, intersectRight, intersectBottom);
                        int image1YOffset = getImageTopLeftY(intersectLeft,
                                intersectTop, intersectRight, intersectBottom);
                        int image2XOffset = t.tileSetX[tileIndex]
                                + (intersectLeft - cellLeft);
                        int image2YOffset = t.tileSetY[tileIndex]
                                + (intersectTop - cellTop);
                        if (doPixelCollision(image1XOffset, image1YOffset,
                                image2XOffset, image2YOffset, this.sourceImage,
                                this.t_currentTransformation, t.sourceImage,
                                TRANS_NONE, intersectWidth, intersectHeight)) {
                            return true;
                        }
                    }
                }
            }
            return false;
        }
    }

    public final boolean collidesWith(Bitmap image, int x, int y,
            boolean pixelLevel) {
        if (!(this.visible)) {
            return false;
        }
        int otherLeft = x;
        int otherTop = y;
        int otherRight = x + image.getWidth();
```

```
        int otherBottom = y + image.getHeight();
        int left = this.x + this.t_collisionRectX;
        int top = this.y + this.t_collisionRectY;
        int right = left + this.t_collisionRectWidth;
        int bottom = top + this.t_collisionRectHeight;
        if (intersectRect(otherLeft, otherTop, otherRight, otherBottom, left,
                    top, right, bottom)) {
            if (pixelLevel) {
                if (this.t_collisionRectX < 0) {
                    left = this.x;
                }
                if (this.t_collisionRectY < 0) {
                    top = this.y;
                }
                if ((this.t_collisionRectX + this.t_collisionRectWidth) > this.width) {
                    right = this.x + this.width;
                }
                if ((this.t_collisionRectY + this.t_collisionRectHeight) > this.height) {
                    bottom = this.y + this.height;
                }
                if (!intersectRect(otherLeft, otherTop, otherRight,
                            otherBottom, left, top, right, bottom)) {
                    return false;
                }
                int intersectLeft = (left < otherLeft) ? otherLeft : left;
                int intersectTop = (top < otherTop) ? otherTop : top;
                int intersectRight = (right < otherRight) ? right : otherRight;
                int intersectBottom = (bottom < otherBottom) ? bottom
                            : otherBottom;
                int intersectWidth = Math.abs(intersectRight - intersectLeft);
                int intersectHeight = Math.abs(intersectBottom - intersectTop);
                int thisImageXOffset = getImageTopLeftX(intersectLeft,
                            intersectTop, intersectRight, intersectBottom);
                int thisImageYOffset = getImageTopLeftY(intersectLeft,
                            intersectTop, intersectRight, intersectBottom);
                int otherImageXOffset = intersectLeft - x;
                int otherImageYOffset = intersectTop - y;
                return doPixelCollision(thisImageXOffset, thisImageYOffset,
                            otherImageXOffset, otherImageYOffset, this.sourceImage,
                            this.t_currentTransformation, image, SpriteCH.TRANS_NONE,
                            intersectWidth, intersectHeight);
            } else {
                return true;
            }
        }
        return false;
    }
```

7.3.3 绘制对话界面

魔塔游戏中的主角可以和 NPC 进行对话，通过对话可以获得对游戏有帮助的信息。对话界面如图 7-2 所示。

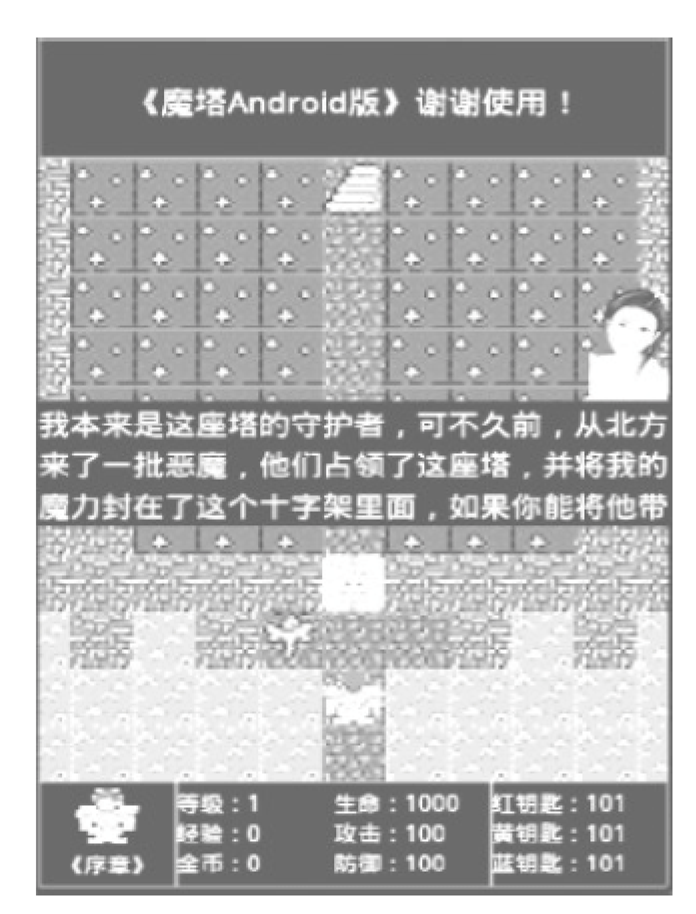


图 7-2　对话界面

图 7-2 所示的界面是通过一个浮动的对话框来显示对话内容，这个对话框只是一个矩形框，然后在右边绘制出对应的 NPC 的头像即可。这里的对话内容可以通过前面介绍的 TextCH 类来实现自动换行。对应代码如下所示。

```
public void dialog()
{
    int x, y, w, h;
    w = yarin.SCREENW;
    h = yarin.MessageBoxH;
    x = 0;
    y = (yarin.SCREENH - yarin.MessageBoxH) / 2;
    if (task.curTask2 % 2 == 0)
    {
        drawDialogBox(IMAGE_DIALOG_HERO, x, y, w, h);
    }
    else
    {
        drawDialogBox(curDialogImg, x, y, w, h);
    }
    tu.DrawText(mcanvas);
}
private void drawDialogBox(int imgType, int x, int y, int w, int h)
{
    Paint ptmPaint = new Paint();
    ptmPaint.setARGB(255,Color.red(BACK_COLOR), Color.green(BACK_COLOR), Color.
    blue(BACK_COLOR));
    yarin.fillRect(mcanvas, x, y, w, h, ptmPaint);
    Bitmap img = getImage(imgType);
    yarin.drawRect(mcanvas, x, y, w, h, ptmPaint);
```

```
            if (img != null)
            {
                if (imgType == IMAGE_DIALOG_HERO)
                {
                    yarin.drawImage(mcanvas, img, x, y - 64);
                }
                else
                {
                    yarin.drawImage(mcanvas, img, yarin.SCREENW - 40, y - 64);
                }
            }
            ptmPaint = null;
        }
```

7.3.4　战斗界面

当主角和怪物发生碰撞时就会发生战斗，在项目中需要专门设计一个界面来实现战斗效果。本项目的战斗界面功能是通过文件 FightScreen.java 实现的，通过此文件可以分别显示玩家和怪物的头像以及属性，包括生命、攻击和防御，主要代码如下所示。

```
        protected void onDraw(Canvas canvas)
        {
            mcanvas = canvas;
            int tx, ty, tw, th;
            tw = yarin.SCREENW;
            th = yarin.MessageBoxH;
            tx = 0;
            ty = (yarin.SCREENH - yarin.MessageBoxH) / 2;
            showMessage();
            if (!isFighting)
            {
                tu.DrawText(mcanvas);
            }
            else
            {
                yarin.drawImage(canvas, orgeImage, 0, ty + (th - GameMap.TILE_WIDTH) / 2, GameMap.
TILE_WIDTH, GameMap.TILE_WIDTH, orgeSrcX, orgeSrcY);
                yarin.drawImage(canvas, heroImage, (tw - GameMap.TILE_WIDTH), ty + (th - GameMap.
TILE_WIDTH) / 2, GameMap.TILE_WIDTH, GameMap.TILE_WIDTH, 0, 0);
                paint.setColor(Color.WHITE);
                //怪物
                {
                    tx = 40;
                    ty = (yarin.SCREENH - yarin.MessageBoxH) / 2 + 5;
                    yarin.drawString(canvas, "生命:" + orgeHp, tx, ty, paint);
                    yarin.drawString(canvas, "攻击:" + orgeAttack, tx, ty + yarin.TextSize, paint);
                    yarin.drawString(canvas, "防御:" + orgeDefend, tx, ty + 2 * yarin.TextSize, paint);
                }
                //英雄
                {
                    String string = "";
```

```
				ty = (yarin.SCREENH - yarin.MessageBoxH) / 2 + 5;
				string = hero.getHp() + ":生命";
				yarin.drawString(canvas, string, (tw - 40 - paint.measureText(string)), ty, paint);
				string = hero.getAttack() + ":攻击";
				yarin.drawString(canvas, string, (tw - 40 - paint.measureText(string)), ty + yarin.TextSize,
paint);
				string = hero.getDefend() + ":防御";
				yarin.drawString(canvas, string, (tw - 40 - paint.measureText(string)), ty + 2 * yarin.
TextSize, paint);
			}
		}
		tick();
	}

	public void showMessage()
	{
		int x = 0;
		int y = (yarin.SCREENH - yarin.MessageBoxH) / 2;
		int w = yarin.SCREENW;
		int h = yarin.MessageBoxH;
		Paint ptmPaint = new Paint();
		ptmPaint.setARGB(255, 0, 0, 0);
		yarin.fillRect(mcanvas, x, y, w, h, ptmPaint);
		ptmPaint = null;
	}

	private void tick()
	{
		if (orgeHp <= 0)
		{
			isFighting = false;
			tu.InitText("得到" + orgeMoney + "个金币 " + "经验值增加" + orgeExperience, 0, (yarin.
SCREENH - yarin.MessageBoxH) / 2, yarin.SCREENW, yarin.MessageBoxH,
				0x0, 0xff0000, 255, yarin.TextSize);
		}
		else if (heroFirst == true)
		{
			orgeHp -= orgeDamagePerBout;
			if (orgeHp <= 0)
			{
				orgeHp = 0;
			}
		}
		else
		{
			hero.cutHp(heroDamagePerBout);
		}
		heroFirst = !heroFirst;
	}

	public boolean onKeyUp(int keyCode)
```

```
{
    switch (keyCode)
    {
        case yarin.KEY_DPAD_UP:
            break;
        case yarin.KEY_DPAD_DOWN:
            break;
        case yarin.KEY_DPAD_OK:
            if (!isFighting)
            {
                hero.addMoney(orgeMoney);
                hero.addExperience(orgeExperience);
                gameScreen.mshowFight = false;
                gameScreen.mFightScreen = null;
                System.gc();
            }
            break;
        case yarin.KEY_SOFT_RIGHT:
            break;
    }
    return true;
}
```

7.3.5　图层管理器

编写文件 LayerCH.java 定义图层管理器类 Layer，前面使用的类 TiledCH 和 SpriteCH 都继承自一个抽象类 Layer（图层）。也就是说，不管是地图还是主角，都包含在图层类中，所以为了方便管理和维护这些图层，构建一个专门用来管理图层的图层管理器 ManagerCH。抽象类 LayerCH 的实现很简单，只是包括了图层的位置（x，y）、图层的宽度和高度（width，height）以及一个控制是否显示图层的布尔变量 visible。

文件 LayerCH.java 的具体实现流程如下所示。

（1）设置并获得图层的一些属性，对应代码如下所示。

```
public abstract class LayerCH {
    //位置和宽度
    int x; // = 0;
    int y; // = 0;
    int width; // = 0;
    int height; // = 0;
    //是否显示
    boolean visible = true;
    //构造方法（宽度，高度）
    LayerCH(int width, int height) {
        setWidthImpl(width);
        setHeightImpl(height);
    }
    //设置位置
    public void setPosition(int x, int y) {
        this.x = x;
        this.y = y;
    }
```

```
        //移动图层
        public void move(int dx, int dy) {
            x += dx;
            y += dy;
        }
        //得到 x
        public final int getX() {
            return x;
        }
        //得到 y
        public final int getY() {
            return y;
        }
        //得到宽度
        public final int getWidth() {
        return width;
        }
        //得到高度
        public final int getHeight() {
        return height;
        }
        //设置是否显示
        public void setVisible(boolean visible) {
           this.visible = visible;
        }
        //得到是否显示
        public final boolean isVisible() {
            return visible;
        }
        //绘制图层的一个抽象方法
        public abstract void paint(Canvas canvas);
        //设置宽度
        void setWidthImpl(int width) {
            if (width < 0) {
                throw new IllegalArgumentException();
            }
           this.width = width;
        }
        //设置高度
        void setHeightImpl(int height) {
            if (height < 0) {
                throw new IllegalArgumentException();
            }
            this.height = height;
        }
    }
```

（2）接下来创建一个图层管理器 ManagerCH。在具体实现时，需要先确定图层管理器需要的成员变量，用视窗的 x 和 y 表示宽度和高度，用一个 Layer 的数组保存所有的图层，用一个变量保存实际图层的数量。在此只需要将所有图层一起添加到图层管理器中，然后设置视图查看时的位置及大小，调用图层管理器的 paint()方法绘制图层。绘制的顺序是按添加的反顺序，即先添加的后绘制，大家一定要注意这一点，以免图

层被覆盖之后显示不出来。上述功能的对应代码如下所示。

```
public class ManagerCH {
    public ManagerCH() {
        setViewWindow(0, 0, Integer.MAX_VALUE, Integer.MAX_VALUE);
    }
    //添加一个图层
    public void append(Layer l) {
        removeImpl(l);
        addImpl(l, nlayers);
    }
    //在指定索引处设置图层
    public void insert(Layer l, int index) {
        if ((index < 0) || (index > nlayers)) {
            throw new IndexOutOfBoundsException();
        }
        removeImpl(l);
        addImpl(l, index)
    }
    //得到指定索引的图层
    public Layer getLayerAt(int index) {
        if ((index < 0) || (index >= nlayers)) {
            throw new IndexOutOfBoundsException();
        }
        return component[index];
    }
    //得到图层的数量
    public int getSize() {
        return nlayers;
    }
    //删除指定的图层
    public void remove(Layer l) {
        removeImpl(l);
    }
    //在指定位置绘制所有的图层
    public void paint(Canvas canvas, int x, int y) {
        canvas.translate(x - viewX, y - viewY);
        //设置裁剪区域
        canvas.clipRect(viewX, viewY, viewX+viewWidth, viewY+viewHeight);
        for (int i = nlayers; --i >= 0;) {
            Layer comp = component[i];
            if (comp.visible) {
                comp.paint(canvas);
            }
        }
        canvas.restore();
        canvas.translate(-x + viewX, -y + viewY);
    }
    //设置视图显示的位置
    public void setViewWindow(int x, int y, int width, int height) {
        if (width < 0 || height < 0) {
            throw new IllegalArgumentException();
```

```
            }
            viewX = x;
            viewY = y;
            viewWidth = width;
            viewHeight = height;
        }
        //添加一个图层（图层，索引）
        private void addImpl(Layer layer, int index) {
            if (nlayers == component.length) {
                Layer newcomponents[ ] = new Layer[nlayers + 4];
                System.arraycopy(component, 0, newcomponents, 0, nlayers);
                System.arraycopy(component, index, newcomponents, index + 1,
                nlayers - index);
                component = newcomponents;
            } else {
                System.arraycopy(component,index,component, index + 1, nlayers
                - index);
            }
            component[index] = layer;
            nlayers++;
        }
      //删除一个指定图层
        private void removeImpl(Layer l) {
            if (l == null) {
                throw new NullPointerException();
            }
            for (int i = nlayers; --i >= 0;) {
                if (component[i] == l) {
                    remove(i);
                }
            }
        }
        //删除一个指定索引的图层
        private void remove(int index) {
            System.arraycopy(component, index + 1, component, index, nlayers
            - index - 1);
            component[--nlayers] = null;
        }
        //实际的图层数
        private int nlayers; // = 0;
        //这里考虑性能的优化，最多设置了 4 层
        private Layer component[ ] = new Layer[4];
        //窗口视图（x,y,w,h）
        private int viewX, viewY, viewWidth, viewHeight; // = 0;
    }
```

7.4　实现游戏音效

知识点讲解：光盘:视频\视频讲解\第 7 章\实现游戏音效.avi

音效在游戏开发中占据重要地位，但开发者在开发游戏时往往把主要精力花费在游戏的图像和动画等方面，而忽视了背景音乐和声音效果，这种做法是不正确的，因为好的游戏音效和音乐可以使玩家融入游戏世界，产生共鸣。本游戏项目中加入了两个背景音乐，一个是菜单背景音乐，一个是游戏中的背景音乐。

（1）首先准备两个符合游戏剧情的背景音乐，放到 res/raw 目录下。

（2）在文件 PlayerCH.java 中创建一个 PlayerCH 类来控制音乐播放，在里面构建了一个 MediaPlayer 对象，通过方法 create()来装载准备好的音乐素材文件。对应代码如下所示。

```
public class PlayerCH
{
public MediaPlayer playerMusic;
public MagicTower magicTower    = null;
public PlayerCH(MagicTower magicTower)
{
    this.magicTower = magicTower;
}
//播放音乐
public void PlayMusic(int ID)
{
    FreeMusic();
    switch (ID)
    {
    case 1:
        //装载音乐
        playerMusic = MediaPlayer.create(magicTower, R.raw.menu);
        //设置循环
        playerMusic.setLooping(true);
        try
        {
        //准备
        playerMusic.prepare();
        }
        catch (IllegalStateException e)
        {
        e.printStackTrace();
        }
        catch (IOException e)
        {
        e.printStackTrace();
        }
        //开始
        playerMusic.start();
        break;
    case 2:
        playerMusic = MediaPlayer.create(magicTower, R.raw.run);
        playerMusic.setLooping(true);
        try
```

```
                {
                playerMusic.prepare();
                }
                catch (IllegalStateException e)
                {
                e.printStackTrace();
                }
                catch (IOException e)
                {
                e.printStackTrace();
                }
                playerMusic.start();
                break;
                }
    }
    //退出释放资源
    public void FreeMusic()
    {
        if (playerMusic != null)
        {
            playerMusic.stop();
            playerMusic.release();
        }
    }
    //停止播放
    public void StopMusic()
    {
        if (playerMusic != null)
        {
            playerMusic.stop();
        }
    }
    }
```

到此为止，整个实例的主要模块代码介绍完毕。本魔塔游戏执行后的界面效果如图 7-3 所示。为节省本书篇幅，其他部分的代码不再详细讲解，读者可以阅读本书附带光盘中对应的源代码。



图 7-3　执行效果

第 8 章　NBA 激情投篮

篮球竞技游戏是以篮球作为主题的游戏，目前主要分为手机游戏、网页游戏、电视游戏、电脑游戏四大类，用户可以扮演球员角色，以操作类游戏为主。随着 Android 篮球手机游戏用户增多，开发一款此类游戏很有必要。本章将通过一个综合篮球游戏实例的实现过程，讲解 Android 技术在游戏项目中的应用流程。在本实例的讲解过程中，始终用科学严谨的语言进行描述，详细讲解了每个知识点，并对难点和重点部分进行了详细剖析。虽然代码繁多，相信经过书中内容和随书光盘中的介绍，读者会一读便懂，使读者真正步入 Android 高手之列。

8.1　篮球游戏介绍

知识点讲解：光盘：视频\视频讲解\第 8 章\篮球游戏介绍.avi

篮球游戏的代表作品包括兄弟篮球、街头篮球和范特西篮球经理等系列。不同类型的篮球游戏，可以让玩家得到不同的体验。篮球游戏平台包括 PSP、PS3、 PS2、NDSL、 PC、手机、XBOX360 等。例如，屏幕视图、道具视图、角色视图等。

8.1.1　篮球游戏介绍

NBA 的全称是 National Basketball Association，是美国第一大职业篮球赛事，产生了迈克尔・乔丹、埃尔文・约翰逊、科比・布莱恩特、姚明、勒布朗・詹姆斯等世界巨星。该协会一共拥有 30 支球队，分属两个联盟：东部联盟和西部联盟。而每个联盟各由 3 个赛区组成，每个赛区有 5 支球队。

自从赛事诞生那天起，就吸引了全世界球迷们的关注。无论是在楼下拍球的篮球少年，还是已经在职业联赛中崭露头角的运动员，都以成为一名 NBA 球员为自己的梦想。为了让更多篮球爱好者体验 NBA 比赛的刺激，开发了激情投篮游戏，和 NBA 巨星们一起进行比赛。

8.1.2　游戏策划

1. 本款游戏的意义

随着工作日益繁忙，人们急切需要通过参加体育活动来放松自己。激情投篮游戏作为一款体育类游戏，模拟了现实世界的体育活动，只要带着手机，玩家便可以随时随地进入虚拟体育世界，享受体育带来的乐趣，满足玩家对体育娱乐性的要求。本游戏的可玩性强，需要玩家在规定时间内得尽可能多的分数。游戏过程中玩家不仅需要控制投篮的方向，还要恰到好处地掌握好投篮的力度，只有协调好这两个因素才能将球顺利投进篮筐，这也是游戏的魅力所在。

开发这款游戏的目的是为读者在 Android 平台上的游戏开发提供一个指导方案，而不是生产商业化的游戏产品。读者可以在这个实例的基础上继续扩充功能，开发出效果更好、更具可玩性的篮球类游戏。

2. 游戏界面分析和技术分析

本程序由如下界面构成：声音设置界面、开始菜单界面、帮助界面、关于界面、加载界面及结束界面。这些界面都是用 2D 技术实现的，均继承和扩展自 SurfaceView，并重写了其中的 onDraw()方法，所有的按钮均为笔者用贴图实现，避免了使用 Android 的自带控件，使游戏画面更加具有视觉冲击力。

游戏试玩界面是使用 3D 技术实现的，游戏界面继承和扩展了 GLSurfaceView 并实现了 Renderer 接口。为了实现界面和操作用户的交互，专门重写了方法 onTouchEvent()。

8.1.3 策划游戏

本项目属于体育类游戏，下面开始策划整个项目的具体功能。

（1）情节

作为一个竞技篮球项目，需要模拟现实世界的篮球投篮实况，所以投篮情节几乎是一模一样的。在此阶段的主要工作是规划游戏进程和不同的场景。

（2）目标用户

本项目主要针对对篮球有一定了解或感兴趣的用户，并且以年轻人为主。

（3）运行平台

本项目的运行平台是 Android 2.3。

（4）显示技术

为了将篮球场场景生动地展示在用户面前，需要采用 2D 单屏模式以指定的视角展示游戏。

（5）操控方式

使用手机键来控制本游戏。

8.1.4 准备工作

在进行游戏开发之前，需要准备好游戏中用到的图片素材和配音文件。其中用到的图片素材文件保存在 res/drawable-mdpi 目录下，如图 8-1 所示。

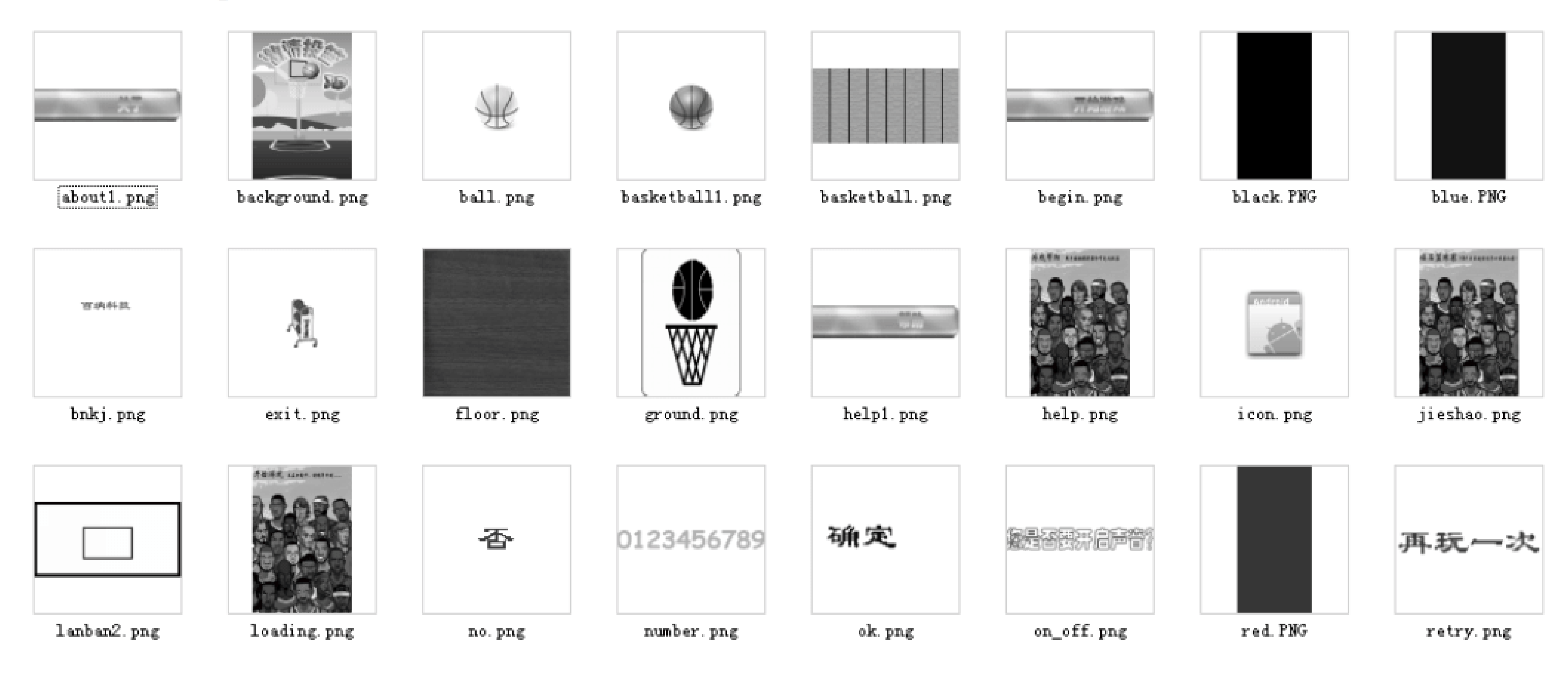


图 8-1 图片素材

用到的配音文件保存在 res/raw 目录下，如图 8-2 所示。



图 8-2　配音文件

8.2　项 目 架 构

知识点讲解：光盘:视频\视频讲解\第 8 章\项目架构.avi

在本节内容中，将对整个项目进行总体架构分析，并对项目中的各个类及其结构进行一一介绍。

8.2.1　总体架构

本项目的总体架构如图 8-3 所示。

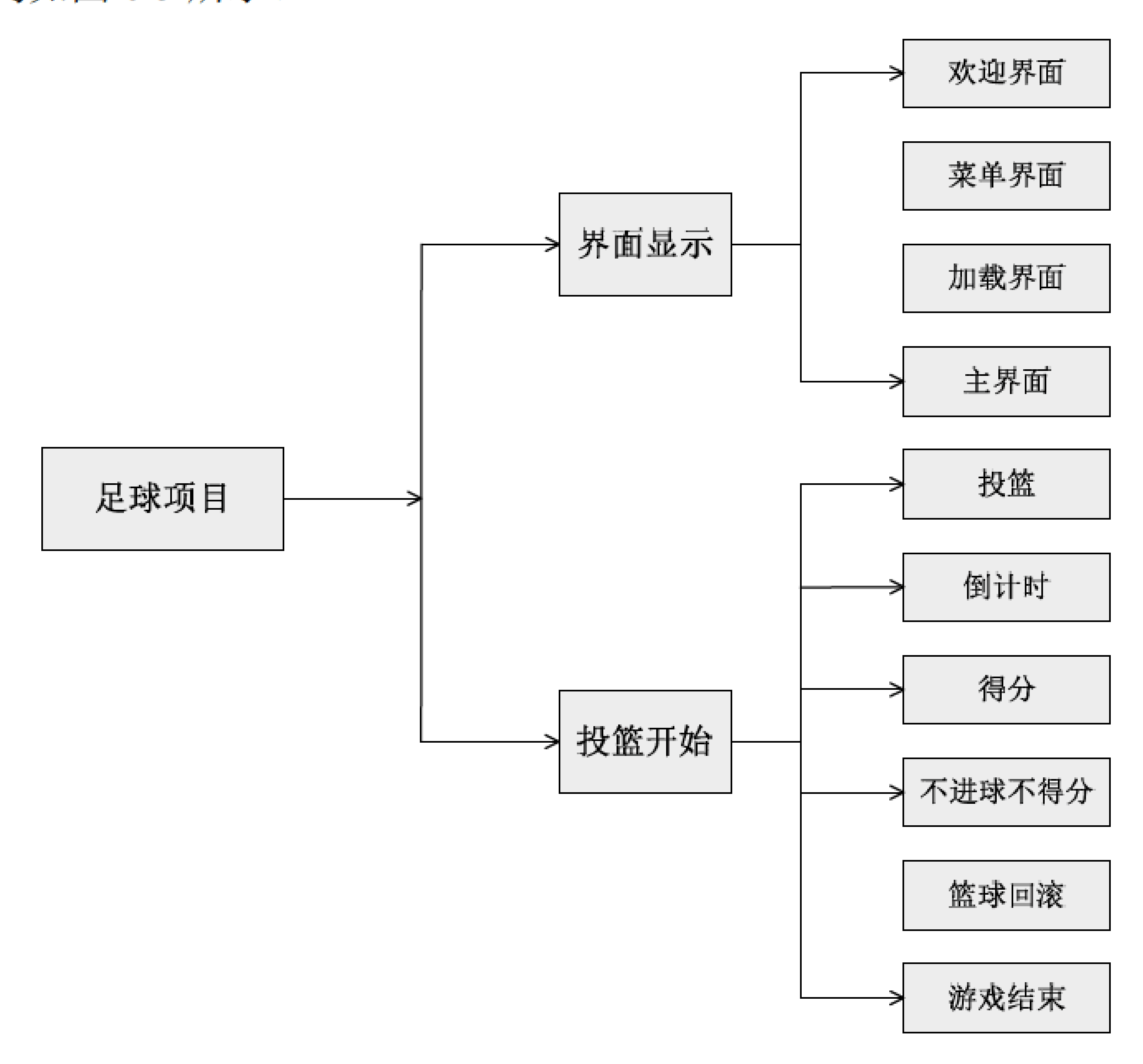


图 8-3　总体架构图

8.2.2　规划类

类是面向对象的核心，在本游戏项目中，需要编写各个类来实现具体的功能。下面将简单介绍各个类的具体功能。

1. 公共类

（1）主类 LanqiuActivity

该类是通过继承和扩展基类 Activity 来实现的，是整个应用程序的入口。主要根据收到的 Handler 消息

的不同切换到不同的界面。

（2）常量类 changL

该类记录程序中所有类中需要用到的常量，便于以后调试修改。

2. 声音设置、开始菜单、帮助、关于、加载及结束界面类

（1）设置声音界面类 Sheng

该类是声音设置界面的实现类，主要负责声音设置界面的绘制，作用是设置游戏中的声音开关。

（2）开始菜单界面类 ZhuView

该类是开始菜单界面的实现类，主要负责开始菜单界面的绘制，是连接整个程序各类的一个集中地，作用是根据单击不同的按钮向 LanqiuActivity 发送不同的 Handler 消息。

（3）帮助界面类 HelpView

该类是游戏帮助界面的实现类，主要负责游戏帮助界面的绘制，作用是显示游戏的规则、操作等信息。

（4）关于界面类 AboutView

该类是游戏关于界面的实现类，主要负责游戏关于界面的绘制，作用是显示游戏相关信息。

（5）加载界面类 LoadView

该类是游戏加载界面的实现类，主要负责游戏加载界面的绘制，作用是承接游戏界面。

（6）游戏结束界面类 OverView

该类是游戏结束界面的实现类，主要负责游戏结束界面的绘制，作用是显示玩家的得分情况和提供再玩一次的功能。

（7）开始菜单线程类 ZhuThread

该类是绘制开始菜单的线程，主要负责绘制开始菜单。

3. 游戏界面类

（1）游戏界面类 GLGameView

该类是游戏中最主要的一个类，游戏规则、游戏模型都包含在此类中。同时该类还负责绘制游戏画面，接收玩家的响应。

（2）碰撞检测类 CollisionUtil

该类是游戏中的一个难点，主要负责碰撞检测，游戏中是否进球是通过此类来判断的。

4. 游戏界面背景类

（1）场地后墙类 Back

该类是通过构造一个贴有墙面纹理的矩形来构造场地后墙的。

（2）场地左墙的构造类 Left

该类是通过构造一个贴有墙面纹理的矩形来构造场地左墙的。

（3）场地右墙类 Right

该类是通过构造一个贴有墙面纹理的矩形来构造场地右墙的。

（4）场地地板类 Floor

该类是通过构造一个矩形贴图来构造场地地板的。

（5）场地屋顶类 Roof

该类是通过构造一个矩形贴图来构造场地屋顶的。

（6）篮板类 Board

该类是通过构造一个贴有篮板纹理的立方体来构造篮板的。



（7）篮环类 Ring

该类是通过对一个圆环贴图来构造篮环的。

（8）支架类 Cylinder

该类是通过对一个圆柱体贴图来构造篮支架的。

（9）篮板、篮环及篮支架的组装类 Assemble

该类是通过对 Board 类、Ring 类及 Cylinder 类的引用，进行合理的平移和旋转来组装篮板的。

（10）绘制仪表板类 Panel

该类是通过接受不同的纹理 ID 和纹理数组绘制不同的仪表板的。

（11）绘制得分类 Score

该类是根据存在变量里的玩家得分值，计算得到相应的得分纹理数组，通过对 Panel 类的引用绘制相应得分的。

（12）倒计时类 Daoji

该类是根据存在变量里的游戏倒计时秒数值，计算得到相应的倒计时秒数纹理数组，通过对 Panel 类的引用绘制相应的倒计时秒数的。

（13）篮球类 BallForDraw

该类是通过构造一个贴有篮球纹理的球来构造篮球的，是真正绘制篮球的类。

8.3　具体编码

知识点讲解：光盘:视频\视频讲解\第 8 章\具体编码.avi

经过前面的讲解，整个项目的前期工作结束。从本节的内容开始，将进入具体编码阶段。希望读者仔细品味本节的内容，真正步入 Android 开发高手的行列。

（1）编写主类 LanqiuActivity

主类 LanqiuActivity 是通过继承和扩展基类 Activity 来实现的，是整个应用程序的入口。主要根据收到的 Handler 消息的不同切换到不同的界面。本篮球游戏项目主类 LanqiuActivity 是由文件 LanqiuActivity.java 实现的，主要实现代码如下所示。

```
package nba.bs;

import java.util.HashMap;

import nba.bs.R;


import android.app.Activity;
import android.content.Context;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.SoundPool;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.Window;
import android.view.WindowManager;
```

```
import static nba.bs.Constant.*;

public class LanqiuActivity extends Activity {
        private GLGameView gameplay;
        public MenuView gamemenu;
        //关于界面
        private AboutView gameabout;
        //帮助界面
        private HelpView gamehelp;
        //结束界面
        private OverView gameover;
        //音效界面
        private Sheng gamesound;
        Handler hd;

        MediaPlayer mpBack;       //游戏背景音乐
        SoundPool soundPool;      //声音
        HashMap<Integer,Integer> soundPoolMap;

       @Override
       protected void onCreate(Bundle savedInstanceState) {
             super.onCreate(savedInstanceState);

             requestWindowFeature(Window.FEATURE_NO_TITLE);
               getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN ,
                                 WindowManager.LayoutParams.FLAG_FULLSCREEN);

               setContentView(R.layout.main);
               new Thread()
               {
                       public void run()
                       {
                               try{
                                       sleep(3000);
                               }
                               catch(Exception e)
                               {
                                       e.printStackTrace();
                               }
                               hd.sendEmptyMessage(GAME_SOUND);
                       }
               }.start();
               initSounds();
             hd=new Handler()//消息处理器初始化
               {
                       @Override
                       public void handleMessage(Message msg)
                       {
                               super.handleMessage(msg);
                               switch(msg.what)
                       {
```

```
case GAME_SOUND:
    gamesound=new Sheng(LanqiuActivity.this);
    setContentView(gamesound);
    break;
case GAME_MENU:
    MENU_FLAG=true;//设置主 Thread 标志位
    gamemenu=new MenuView(LanqiuActivity.this);
    setContentView(gamemenu);
    break;
case GAME_LOAD:
    MENU_FLAG=false;//设置主 Thread 标志位
    setContentView(R.layout.loading);
    new Thread()
    {
        public void run()
        {
            try{
                sleep(2000);
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
            hd.sendEmptyMessage(GAME_PLAY);
        }
    }.start();

    //this.sendEmptyMessage();
    break;
    case GAME_PLAY://处理游戏信息
    score=0;//还原得分
    daojis=60;//还原倒计时
    SOUND_FLAG=SOUND_MEMORY;//还原声音
    DAOJI_FLAG=true;//打开倒计时
    new DaojiThread(LanqiuActivity.this).start();//开启倒计时

    gameplay = new GLGameView(LanqiuActivity.this,gamemenu);
    gameplay.requestFocus();
    gameplay.setFocusableInTouchMode(true);
    if(SOUND_FLAG)
    {
        mpBack.setLooping(true);
        mpBack.setVolume(0.2f, 0.2f);
        mpBack.start();
    }
    setContentView(gameplay);
    break;
    case GAME_ABOUT:
    MENU_FLAG=false;//设置主 Thread 标志位
    gameabout=new AboutView(LanqiuActivity.this);
    setContentView(gameabout);
```

```
                    break;
                case GAME_HELP:
                    MENU_FLAG=false;//设置主 Thread 标志位
                    gamehelp=new HelpView(LanqiuActivity.this);
                    setContentView(gamehelp);
                    break;
                case GAME_OVER:
                    gameover=new OverView(LanqiuActivity.this,gamemenu);
                    setContentView(gameover);
                    break;
                case RETRY:
                    gamemenu=new MenuView(LanqiuActivity.this);
                    setContentView(gamemenu);
                    break;
                }
            }
        };
    }
    //重新播放背景音乐
    protected void onResume() {
        super.onResume();
        if(gameplay!=null)
        {
          gameplay.onResume();
           mpBack.start();
        }

    }
    //暂停背景音乐
    @Override
    protected void onPause() {
        super.onPause();
        gameplay.onPause();
        mpBack.pause();
    }

    public void initSounds()
    {
     mpBack=MediaPlayer.create(this,R.raw.gameback);
     //在此可以编写需要的 MediaPlayer
     soundPool=new SoundPool
     (
          4,
          AudioManager.STREAM_MUSIC,
          100
     );
     soundPoolMap=new HashMap<Integer,Integer>();
     soundPoolMap.put(1, soundPool.load(this,R.raw.collision,1));
     soundPoolMap.put(2, soundPool.load(this,R.raw.over,1));
    }
    //播放背景音乐
```

```
public void playSound(int sound,int loop)
{
 AudioManager mgr=(AudioManager)this.getSystemService(Context.AUDIO_SERVICE);
//现在的音量
 float streamVolumeCurrent=mgr.getStreamVolume(AudioManager.STREAM_MUSIC);
//最大音量
 float streamVolumeMax=mgr.getStreamMaxVolume(AudioManager.STREAM_MUSIC);

 float volume=streamVolumeCurrent/streamVolumeMax;

 soundPool.play(soundPoolMap.get(sound), volume, volume, 1, loop, 0.5f);
}
```

（2）编写常量类

很多初级程序员常常会忽略在应用程序中编写常量类的作用。建议在开发项目时，把应用程序中要用的所有常量集中写到一个类中。这样不但有利于调试程序时查找，而且还可避免因数字太多引起的混淆。本实例的常量类在文件 changL.java 中实现，具体代码如下所示。

```
public class changL{
    //场地大小，长、宽、高
    public static float WIDTH=4.1f;
    public static float HEIGHT=7f;
    public static float LENGTH=4f;
    //篮球大小
    public static final float BALL_ANGLESPAN=15f;
    public static final float BALL_SCALE=0.3f;
    //场地纹理数组
    public static float[ ] HALL_TEXTURES=new float[ ]{
            0,0,0,1.0f,1.0f,1.0f,
            0,0,1.0f,1.0f,1.0f,0
    };
    //重力加速度
    public static float G=0.8f;

    public static float CAMERA_INI_X=0;
    public static float CAMERA_INI_Y=HEIGHT/2;
    public static float CAMERA_INI_Z=LENGTH+4.0f;

    public static final float DISTANCE=LENGTH;

    public static final float ENERGY_LOSS=0.4f;

    public static final float BALL_MAX_SPEED_X=0.6f;
    public static final float BALL_MAX_SPEED_Y=3.0f;
    public static final float BALL_MAX_SPEED_Z=-2.0f;

    public static final float BALL_NEAREST_Z=LENGTH/2-0.15f;//球最近距离

    public static final float BALL_FLY_TIME_SPAN=0.1f;//球飞行的时间
    public static final float BALL_ROLL_SPEED=0.05f;//球在地上的滚动速度
    public static final float BALL_ROLL_ANGLE=360*BALL_ROLL_SPEED/(2*(float)Math.PI*BALL_SCALE);
```

```
    public static final float BASKETBALL_STANDS_SPAN=0.08f;      //篮球架大小比例
    public static final float BASKETBALL_STANDS_X=0;             //篮板 x 坐标
    public static final float BASKETBALL_STANDS_Y=5;             //篮板 y 坐标
    public static final float BASKETBALL_STANDS_Z=-1.65f;        //篮板 z 坐标

    //仪表板中单个数字的大小
    public static final float SCORE_NUMBER_SPAN_X=0.1f;
    public static final float SCORE_NUMBER_SPAN_Y=0.12f;

    public static float[ ] ringCenter;                           //篮环中心点坐标
    public static float ringR;                                   //篮环大小
    public static int score=0;                                   //得分
    public static int daojis=60;                                 //倒计时

    //阴影
    public static float SHADOW_X=1.0f;
    public static float SHADOW_Y=0.1f;
    public static float SHADOW_Z=0.6f;

    //菜单
    public static final int GAME_SOUND=1;
    public static final int GAME_MENU=2;
    public static final int GAME_LOAD=3;
    public static final int GAME_HELP=4;
    public static final int GAME_ABOUT=5;
    public static final int GAME_PLAY=6;
    public static final int GAME_OVER=7;
    public static final int RETRY=8;
    public static float LEFT=-55f;
    public static boolean SOUND_FLAG=true;
    public static boolean SOUND_MEMORY=false;
    public static boolean DAOJI_FLAG=false;
    public static boolean MENU_FLAG=false;
    public static boolean BALL_GO_FLAG=true;
}
```

（3）编写游戏开始菜单类 zhuView

游戏开始菜单是本游戏项目的一个重要界面，当一开始进入游戏，设置完声音后便会进入这个界面。游戏获胜后也会来到这个界面。开始菜单类 zhuView 是本项目最主要的类之一，实现过程比较复杂。本游戏开始菜单类 zhuView 的实现文件是 zhuView.java，主要代码如下所示。

```
public class zhuView extends SurfaceView implements SurfaceHolder.Callback
{
    LanqiuActivity activity;//引用 activity

    Paint paint;
    Bitmap begin;
    Bitmap backgroud;
    Bitmap shut;
    Bitmap about1;
    Bitmap help;
```

```
float left1=LEFT;
float left2=LEFT;
float left3=LEFT;
float left4=LEFT;

public zhuView(LanqiuActivity activity) {
        super(activity);
        this.activity=activity;
        this.getHolder().addCallback(this);             //生命周期回调接口
        //画笔
        paint = new Paint();
        paint.setAntiAlias(true);                       //抗锯齿功能

        initBitmap();//初始化位图
}
//绘制位图
public void onDraw(Canvas canvas)
{
        if(canvas==null) return;
        super.onDraw(canvas);
        canvas.drawBitmap(backgroud, 0, 0, paint);
        canvas.drawBitmap(begin, left1, 300, paint);
        canvas.drawBitmap(help, left2, 340, paint);
        canvas.drawBitmap(about1, left3, 380, paint);
        canvas.drawBitmap(shut, left4, 420, paint);
}
//触屏控制事件
public boolean onTouchEvent(MotionEvent e)
{
        float x=e.getX();
        float y=e.getY();

        switch(e.getAction())
        {
        case MotionEvent.ACTION_DOWN://按下事件
                if(x>=0&&x<=105&&y>=300&&y<=335)
                {
                        left1=LEFT+20;
                        left2=LEFT;
                        left3=LEFT;
                        left4=LEFT;
                }
                else if(x>=0&&x<=105&&y>=340&&y<=375) //帮助
                {
                        left1=LEFT;
                        left2=LEFT+20;
                        left3=LEFT;
                        left4=LEFT;
                }
                else if(x>=0&&x<=105&&y>=380&&y<=415) //关于按钮
```

```
			{
				left1=LEFT;
				left2=LEFT;
				left3=LEFT+20;
				left4=LEFT;
			}
			else if(x>=0&&x<=105&&y>=420&&y<=455)		//退出
			{
				left1=LEFT;
				left2=LEFT;
				left3=LEFT;
				left4=LEFT+20;
			}
			new ZhuThread(this).start();

			break;
			case MotionEvent.ACTION_UP:			//开始
			if(x>=0&&x<=105&&y>=300&&y<=335)
			{
				activity.hd.sendEmptyMessage(GAME_LOAD);
			}
			else if(x>=0&&x<=105&&y>=340&&y<=375)
			{
				activity.hd.sendEmptyMessage(GAME_HELP);
			}
			else if(x>=0&&x<=105&&y>=380&&y<=415)
			{
				activity.hd.sendEmptyMessage(GAME_ABOUT);
			}
			else if(x>=0&&x<=105&&y>=420&&y<=455)
			{
				System.exit(0);
			}
		}
		return true;
	}
	@Override
	public void surfaceChanged(SurfaceHolder holder, int format, int width,
			int height) {
	}

	//创建 view
	public void surfaceCreated(SurfaceHolder holder) {
		Canvas canvas=holder.lockCanvas();
		try
		{
			synchronized(holder)
			{
				onDraw(canvas);//绘制
			}
		}catch(Exception e)
```

```
                {
                        e.printStackTrace();
                }
                finally
                {
                        if(canvas!=null)
                        {
                                holder.unlockCanvasAndPost(canvas);
                        }
                }
        }
        //销毁创建的 view
        public void surfaceDestroyed(SurfaceHolder holder) {
        }
        //位图实例化
        public void initBitmap()
        {
                backgroud=BitmapFactory.decodeResource(activity.getResources(),  R.drawable.background);//背景
初始化
                begin=BitmapFactory.decodeResource(activity.getResources(), R.drawable.begin); //开始按钮初始化
               //退出按钮
                shut=BitmapFactory.decodeResource(activity.getResources(), R.drawable.shut);
               //关于按钮初始化
                about1=BitmapFactory.decodeResource(activity.getResources(), R.drawable.about1);
                help=BitmapFactory.decodeResource(activity.getResources(), R.drawable.help1);       //帮助按钮初
始化
        }
}
```

接下来需要编写开始菜单绘制线程类 zhuThread，实现文件是 zhuThread.java，功能是绘制开始菜单。主要代码如下所示。

```
public class zhuThread extends Thread
{
        zhuView cc;
        SurfaceHolder holder;
        public zhuThread(zhuView cc)
        {
                this.cc=cc;
                this.holder=cc.getHolder();
        }
        public void run()//重写方法
        {
                Canvas canvas;//画布
                while(MENU_FLAG)
                {
                        canvas=null;//清空画布
                        if(true)
                        {
                                try{
                                        canvas=this.holder.lockCanvas();//锁住画布
                                        synchronized(this.holder)
```

```
                    {
                        cc.onDraw(canvas);
                    }
                }catch(Exception e)
                {
                    e.printStackTrace();
                }
                finally
                {
                    if(canvas!=null)
                    {
                        this.holder.unlockCanvasAndPost(canvas);//画布解锁
                    }
                }
            }
            try{
                sleep(200);//200 毫秒绘制一次
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
        }
    }
}
```

注意： 本项目的帮助界面、关于界面等功能非常简单，设计过程和主界面的设计过程类似。为节省本书篇幅，将不再进行讲解，读者参考本书光盘中的源代码即可。

（4）设计游戏试玩界面

本游戏的核心是游戏试玩界面，这是一个 3D 界面，其中继承了 GLSurfaceView。此界面的主要作用是组装各个已经实现了的部件，并判断游戏是否结束。本游戏试玩界面的实现文件是 shiwan.java，具体实现流程如下所示。

实现框架设计，对应代码如下所示。

```
class shiwan extends GLSurfaceView {
    private SceneRenderer mRenderer;

    zhuView w;//声明菜单界面引用

    public int floorTexId;
    public int balltextureid;
    public int sidewallTexId1;
    public int sidewallTexId2;
    public int BackTexId;
    public int roofTexId;
    public int lzjTextureid;
    public int lhTextureid;
    public int lbTextureid;
    public int scorebankid;
    public int numberid;
```

```
    public int shadowid;

    //摄像机位置 x、y、z
    public float cx=CAMERA_INI_X;
    public float cy=CAMERA_INI_Y;
    public float cz=CAMERA_INI_Z;
    //目标点位置
    public float tx=0;
    public float ty=CAMERA_INI_Y;
    public float tz=0;

    public int screenWidth;
    public int screenHeight;

    public float touchStartY;
    public float touchStartX;
    LogicalBall currentTouchBall;

    public shiwan(Context context,zhuView w) {
        super(context);
        mRenderer = new SceneRenderer();                          //创建渲染器
        setRenderer(mRenderer);                                   //设置渲染器
        setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);     //设置为主动渲染

        this.w=w;
   }
    LanqiuActivity activity=(LanqiuActivity)this.getContext();

    //按键回调方法
    @Override
    public boolean onKeyDown(int keyCode,KeyEvent e)
    {
        switch(e.getAction())
        {
        case KeyEvent.ACTION_DOWN:
            if(keyCode==4)
            {
                w.left1=LEFT;//菜单位置
                w.left2=LEFT;
                w.left3=LEFT;
                w.left4=LEFT;

                SOUND_FLAG=false;                          //关闭声音
                DAOJI_FLAG=false;                          //关闭倒计时
                activity.hd.sendEmptyMessage(GAME_MENU);
            }
            break;
        }
        return true;
    }
```

☑ 实现触屏响应处理事件，具体代码如下所示。

```
//触摸事件
public boolean onTouchEvent(MotionEvent e) {
    float y = e.getY();
    float x = e.getX();
    switch (e.getAction())
    {
     case MotionEvent.ACTION_DOWN://按下
       touchStartY = y;          //记录按下横坐标
         touchStartX = x;        //记录按下纵坐标
         //判断是否摸到球
         float x3d=WIDTH*x/screenWidth-0.5f*WIDTH;
         float y3d=HEIGHT*0.9f*(screenHeight-y)/screenHeight;
         for(LogicalBall bfcc:mRenderer.albfc)
          {
              if(bfcc.state==0&&
                x3d<bfcc.currentX+BALL_SCALE&&x3d>bfcc.currentX-BALL_SCALE&&
                y3d<bfcc.currentY+BALL_SCALE&&y3d>bfcc.currentY-BALL_SCALE)
              {
                 currentTouchBall=bfcc;
                 break;
              }
          }
     break;
     case MotionEvent.ACTION_UP://弹起触控
         float dx=x-touchStartX;
         float dy=y-touchStartY;

         if(currentTouchBall!=null)
         {
           float vx=dx*BALL_MAX_SPEED_X/screenWidth;
           float vy=-dy*BALL_MAX_SPEED_Y/screenHeight;
           float vz=dy*BALL_MAX_SPEED_Z/screenHeight;
           currentTouchBall.vx=vx*3;
           currentTouchBall.vy=vy*4;
           currentTouchBall.vz=vz*2;
           currentTouchBall.state=2;

           currentTouchBall=null;
         }
     break;
    }
    return true;
}
```

☑ 定义方法 onDrawFrame()来绘制游戏场景，主要代码如下所示。

```
    public void onDrawFrame(GL10 gl) {
        gl.glShadeModel(GL10.GL_SMOOTH);
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
        gl.glMatrixMode(GL10.GL_MODELVIEW);
        gl.glLoadIdentity();
```

```
		//摄像机参数
		GLU.gluLookAt
		(
				gl,
				cx,
				cy,
				cz,
				tx,
				ty,
				tz,
				0,
				1,
				0
		);


		gl.glPushMatrix();                    //矩阵现场
		floor.drawSelf(gl);                   //场地部件
		Back.drawSelf(gl);                    //场地部件
		Left.drawSelf(gl);                    //场地部件
		Right.drawSelf(gl);                   //场地部件
		roof.drawSelf(gl);                    //场地部件
		basketball_stands.drawSelf(gl);//绘制篮球
		gl.glPopMatrix();

		for(LogicalBall bfcc:albfc)
		{
			bfcc.drawSelf(gl);
		}
	//设置为模式矩阵
		gl.glMatrixMode(GL10.GL_MODELVIEW);
		gl.glLoadIdentity();

		gl.glPushMatrix();
		gl.glTranslatef(0, 1.2f, -3f);
		sb.drawSelf(gl);
		gl.glPopMatrix();

		gl.glEnable(GL10.GL_BLEND);//打开混合
		gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE_MINUS_SRC_ALPHA);
		gl.glPushMatrix();
		gl.glTranslatef(-0.8f, 1.12f, -2.8f);
		score.drawSelf(gl);
		gl.glPopMatrix();

		gl.glPushMatrix();
		gl.glTranslatef(0.55f, 1.12f, -2.8f);
		daoji.drawSelf(gl);
		gl.glPopMatrix();
}
```

☑ 定义方法 onSurfaceCreated()来创建场景，部分代码如下所示。

```
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
  gl.glDisable(GL10.GL_DITHER);
      gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,GL10.GL_FASTEST);
      gl.glClearColor(0,0,0,0);
      gl.glEnable(GL10.GL_DEPTH_TEST);

        floorTexId=initTexture(gl,R.drawable.floor);
        sidewallTexId1=initTexture(gl,R.drawable.swall1);
        sidewallTexId2=initTexture(gl,R.drawable.swall3);
        BackTexId=initTexture(gl,R.drawable.swall2);
        roofTexId=initTexture(gl,R.drawable.black);
        balltextureid=initTexture(gl,R.drawable.basketball);
        lzjTextureid=initTexture(gl,R.drawable.icon);
        lhTextureid=initTexture(gl, R.drawable.red);
        lbTextureid=initTexture(gl, R.drawable.lanban2);
        scorebankid=initTexture(gl,R.drawable.yibiaoban);
        numberid=initTexture(gl,R.drawable.number);

        shadowid=initTexture(gl,R.drawable.shadow);

        sb=new Panel
        (
                    2.2f,
                    0.3f,
                    scorebankid,
                    new float[ ]
                    {
                          0,0,0,1,1,0,
                          1,0,0,1,1,1
                    }
        );
        score=new Score(numberid,shiwan.this);
        daoji=new Daoji(numberid,shiwan.this);

        Back=new Back(BackTexId,WIDTH,HEIGHT,LENGTH);
        floor=new Floor(WIDTH,HEIGHT,LENGTH,floorTexId);
        Left=new Left(sidewallTexId1,WIDTH,HEIGHT,LENGTH);
        Right=new Right(sidewallTexId2,WIDTH,HEIGHT,LENGTH);
        roof=new Roof(roofTexId,WIDTH,HEIGHT,LENGTH);

        ball=new BallForDraw(BALL_ANGLESPAN,BALL_SCALE,balltextureid);

        shadow=new Floor(SHADOW_X,0,SHADOW_Z,shadowid);
        //篮球架
        basketball_stands=new Assemble
        (
                    BASKETBALL_STANDS_SPAN,
                    BASKETBALL_STANDS_X,
                    BASKETBALL_STANDS_Y,
                    BASKETBALL_STANDS_Z,
                    shiwan.this,
```

```
                        lzjTextureid,
                        lhTextureid,
                        lbTextureid
                );
                ringCenter=basketball_stands.getRingCentre();
                ringR=basketball_stands.getRingReduis();

                albfc.add(new LogicalBall(ball,shadow,0,activity));
                albfc.add(new LogicalBall(ball,shadow,-2*BALL_SCALE,activity));
                albfc.add(new LogicalBall(ball,shadow,2*BALL_SCALE,activity));
                bgt=new Ball_Go_Thread(albfc);
                bgt.start();
        }
```

（5）绘制游戏场景

构成本游戏场景的部件有多个，接下来将详细讲解比较典型部件的绘制方法。其他部件的绘制原理和实现方法都比较类似，为节省本书篇幅，将不再详细讲解。

☑　编写文件 Daoji.java 实现倒计时功能，主要代码如下所示。

```
public class Daoji
{
    shiwan mv;
    Panel[ ] numbers=new Panel[10];

    public Daoji(int texId,shiwan mv)
    {
        this.mv=mv;

        //生成 0～9 共 10 个纹理矩形
        for(int i=0;i<10;i++)
        {
            numbers[i]=new Panel
              (
                SCORE_NUMBER_SPAN_X,
                SCORE_NUMBER_SPAN_Y,
                 texId,
                 new float[ ]
                        {
                            0.1f*i,0, 0.1f*i,1, 0.1f*(i+1),0,
                            0.1f*(i+1),0, 0.1f*i,1, 0.1f*(i+1),1
                        }
               );
        }
    }

    public void drawSelf(GL10 gl)
    {
        String scoreStr=daojis+"";
        for(int i=0;i<scoreStr.length();i++)
        {//绘制得分中的数字字符
            char c=scoreStr.charAt(i);
            gl.glPushMatrix();
            gl.glTranslatef(i*SCORE_NUMBER_SPAN_X, 0, 0);
```

```
                numbers[c-'0'].drawSelf(gl);
                gl.glPopMatrix();
            }
        }
}
```

☑ 编写文件 lanban.java 绘制篮板，需要非常复杂的数学模型来绘制一个真实的整体篮板，这些数学模型包括圆环、圆柱和六面立方体。用圆环知识构造篮环，用圆柱知识构造篮筐支架，用六面立方体的知识构建篮板。主要代码如下所示。

```
public class lanban
{
    private Cylinder lzj;           //支架
    private Ring lh;                //篮环
    private Board lb;               //篮板

    float offsetx;
    float offsety;
    float offsetz;

    float zuobiaospan;

    public lanban//构造器
    (float zuobiaospan,float offsetx,float offsety,float offsetz,shiwan mySurface,
                int lzjTextureid,int lhTextureid,int lbTextureid
    )
    {
        this.zuobiaospan=zuobiaospan;
        this.offsetx=offsetx;
        this.offsety=offsety;
        this.offsetz=offsetz;

        lzj=new Cylinder(3*zuobiaospan,zuobiaospan/5,45,10,lzjTextureid);
        lh=new Ring(18,45,6*zuobiaospan,zuobiaospan/5,lhTextureid);
        lb=new Board(zuobiaospan/2,36*zuobiaospan,21*zuobiaospan,lbTextureid);

    }

    public void drawSelf(GL10 gl)
    {
        //绘制篮板
        gl.glPushMatrix();
        gl.glTranslatef(offsetx, offsety, offsetz);
        lb.drawSelf(gl);
        gl.glPopMatrix();

        //绘制支架
        gl.glPushMatrix();
        gl.glTranslatef
        (
                offsetx+lh.ring_Radius/2,
                offsety-lb.height/4,
```

```
                    offsetz+lzj.length/2+lb.length/2
            );
            gl.glRotatef(90, 0, 1, 0);
            lzj.drawSelf(gl);
            gl.glPopMatrix();

            gl.glPushMatrix();
            gl.glTranslatef
            (
                    offsetx-lh.ring_Radius/2,
                    offsety-lb.height/4,
                    offsetz+lzj.length/2+lb.length/2
            );
            gl.glRotatef(90, 0, 1, 0);
            lzj.drawSelf(gl);
            gl.glPopMatrix();

            //绘制篮筐
            gl.glPushMatrix();
            gl.glTranslatef
            (
                    offsetx,
                    offsety-lb.height/4,
                    (float) (offsetz+lb.length/2+lzj.length+Math.sqrt(3)/2*lh.ring_Radius)
            );
            lh.drawSelf(gl);
            gl.glPopMatrix();
    }

    public float[ ] getRingCentre()
    {
            float[ ] ringCentre=new float[ ]
            {
                    offsetx,
                    offsety-lb.height/2,
                    (float) (offsetz+lb.length/2+lzj.length+Math.sqrt(3)/2*lh.ring_Radius)
            };
            return ringCentre;
    }

    public float getRingReduis()
    {
            float ringReduis=lh.ring_Radius;
            return ringReduis;
    }
}
```

☑ 编写文件 yundongBall.java 实现篮球在场地中的运动，本游戏中篮球的运动抛物线完全模拟现实中的篮球运动的抛物线轨迹。具体代码如下所示。

```
public class yundongBall{
    float vx;
```

```
float vy;
float vz;
float timeLive;
float startX;
float startY;
float startZ;
BallForDraw ball;
Floor shadow;
int state;//球的状态，0 表示停止，1 表示在地面上，2 表示飞行中

//当前位置
float currentX;
float currentY;
float currentZ;

//上一时间位置
float previousX;
float previousY;
float previousZ;

float xAngle=0;
LanqiuActivity activity;

public yundongBall(BallForDraw ball,Floor shadow,float startX,LanqiuActivity activity)
{
        this.ball=ball;
        this.shadow=shadow;
        this.startX=startX;
        this.activity=activity;

        startY=0;
        startZ=BALL_NEAREST_Z;
        state=2;

        vx=0f;
        vy=0f;
        vz=0f;

        currentX=startX;
        currentY=startY;
        currentZ=startZ;
}

public void drawSelf(GL10 gl)
{
        gl.glEnable(GL10.GL_BLEND);//打开混合
        gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE_MINUS_SRC_ALPHA);
        gl.glPushMatrix();
        gl.glTranslatef(currentX, SHADOW_Y, currentZ);
```

```
        shadow.drawSelf(gl);
        gl.glPopMatrix();
        gl.glDisable(GL10.GL_BLEND);

        gl.glPushMatrix();
        gl.glTranslatef(0, changL.BALL_SCALE, 0);//到地面
        gl.glTranslatef(currentX, currentY, currentZ);
        gl.glRotatef(xAngle, 1, 0, 0);
        ball.drawSelf(gl);
        gl.glPopMatrix();
    }

    public void move()
    {
        if(state==0)
        {//在地面静止
            currentX=startX;
            currentY=startY;
            currentZ=startZ;
        }
        else if(state==1)
        {//在地面滚动
            if(currentZ<BALL_NEAREST_Z)
            {
                currentZ=currentZ+BALL_ROLL_SPEED;
                xAngle=xAngle+BALL_ROLL_ANGLE;
            }
            else
            {
                currentZ=BALL_NEAREST_Z;

                startX=currentX;
                startY=currentY;
                startZ=currentZ;

                state=0;
            }
        }
        else if(state==2)
        {//在空中飞
            timeLive=timeLive+BALL_FLY_TIME_SPAN;
            float tempCurrentX=startX+vx*timeLive;
            float tempCurrentY=startY+vy*timeLive-0.5f*G*timeLive*timeLive;
            float tempCurrentZ=startZ+vz*timeLive;

            boolean backFlag=false;
            float[ ] ballCenter={tempCurrentX,tempCurrentY,tempCurrentZ};
          //球与篮环的接触点
            float[ ] point=CollisionUtil.breakPoint
```

```
                (
                        ballCenter,
                        BALL_SCALE,
                        ringCenter,
                        ringR

                );

                if(point!=null)
                {//有接触点则为碰撞篮环
                    if(SOUND_FLAG)
                    {
                        activity.playSound(1, 0);
                    }

                    float[ ] vBefore={vx,vy,vz};
                    float[ ] vAfter=CollisionUtil.ballBreak(vBefore, ballCenter, point);

                    vx=vAfter[0]+(float)Math.random()*0.3f;
                    vy=vAfter[1]+0.5f*((currentY>ringCenter[1])?1:-1);
                    vz=vAfter[2];
                    //清零生存期
                    timeLive=0;
                    //重置开始点
                    startX=currentX;
                    startY=currentY;
                    startZ=currentZ;
                    return;
                }
                //判断是否撞到下面或上面
                if(!backFlag&&tempCurrentY<0||tempCurrentY>HEIGHT-BALL_SCALE)
                {
                    //恢复标志
                    backFlag=true;
                    //Y 向速度
                    vy=vy-G*timeLive;
                    vy=-vy;
                    //清零生存期
                    timeLive=0;
                    //设置开始点
                    startX=currentX;
                    startY=currentY;
                    startZ=currentZ;
                }
        //是否撞前面或后面
        if(!backFlag&&tempCurrentZ<-0.5f*LENGTH+BALL_SCALE||tempCurrentZ>BALL_NEAREST_Z)
                {
                    //恢复标志
                    backFlag=true;
```

```
                vz=-vz;
                vy=vy-G*timeLive;
                //清零生存期
                timeLive=0;
                //开始点
                startX=currentX;
                startY=currentY;
                startZ=currentZ;
            }
            //判断是否撞右面或左面
    if(!backFlag&&tempCurrentX>0.5f*WIDTH-BALL_SCALE||tempCurrentX<-0.5f*WIDTH+BALL_SCALE)
            {
                //恢复标志
                backFlag=true;
                vx=-vx;
                vy=vy-G*timeLive;
                //清零生存期
                timeLive=0;
                //重置开始点
                startX=currentX;
                startY=currentY;
                startZ=currentZ;
            }
            if(!backFlag)
            {//如果没有撞上则移动球
                previousX=currentX;
                previousY=currentY;
                previousZ=currentZ;

                currentX=tempCurrentX;
                currentY=tempCurrentY;
                currentZ=tempCurrentZ;
                if(previousY>ringCenter[1]&&currentY<ringCenter[1]&&
                   Math.sqrt
                   (
                           (previousX-ringCenter[0])*(previousX-ringCenter[0])+
                           (previousZ-ringCenter[2])*(previousZ-ringCenter[2])
                   )<ringR &&
                   Math.sqrt
                   (
                           (currentX-ringCenter[0])*(currentX-ringCenter[0])+
                           (currentZ-ringCenter[2])*(currentZ-ringCenter[2])
                   )<ringR
                )
                {
                    score++;
                }
            }
            else
            {//撞上则损耗能量
                if(SOUND_FLAG)
                {
```

```
                        activity.playSound(1, 0);
                    }

                    vx=ENERGY_LOSS*vx;
                    vy=ENERGY_LOSS*vy;
                    vz=ENERGY_LOSS*vz;
                    //速度低于阈值则切换
                    float vTotal=(float)Math.sqrt(vx*vx+vy*vy+vz*vz);
                if(vTotal<0.1f&&currentY<2*BALL_SCALE)
                {
                    vx=0;
                    vy=0;
                    vz=0;
                    state=1;
                }
            }
        }

    }
}
```

☑ 编写文件 Ball_Thread.java，在其中定义类 Ball_Thread 来控制单个球的运动状态。具体代码如下所示。

```
public class Ball_Thread extends Thread
{
    List<yundongBall> albfc;

    public Ball_Thread(List<yundongBall> albfc)
    {
        this.albfc=albfc;
    }

    public void run(){
        while(BALL_GO_FLAG)
        {
            for(yundongBall lb:albfc)
            {//用 for 循环语句控制每一个球
                lb.move();
            }

            try{
                sleep(50);
            }
            catch(Exception e){
                e.printStackTrace();
            }
        }

    }
}
```

到此为止，本游戏项目的主要功能介绍完毕。至于其他模块的功能，为节省本书篇幅就不再进行讲解。

请读者朋友参考本书光盘中的源代码，里面有完整的注释，相信大家一读便懂。

本游戏项目执行之后的载入界面如图 8-4 所示，游戏主界面如图 8-5 所示。

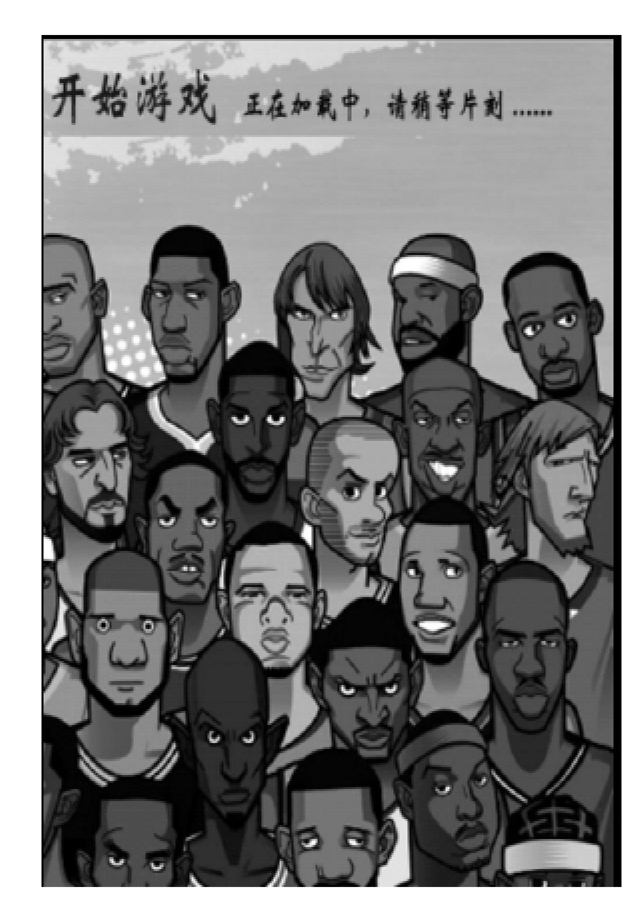


图 8-4　游戏加载界面

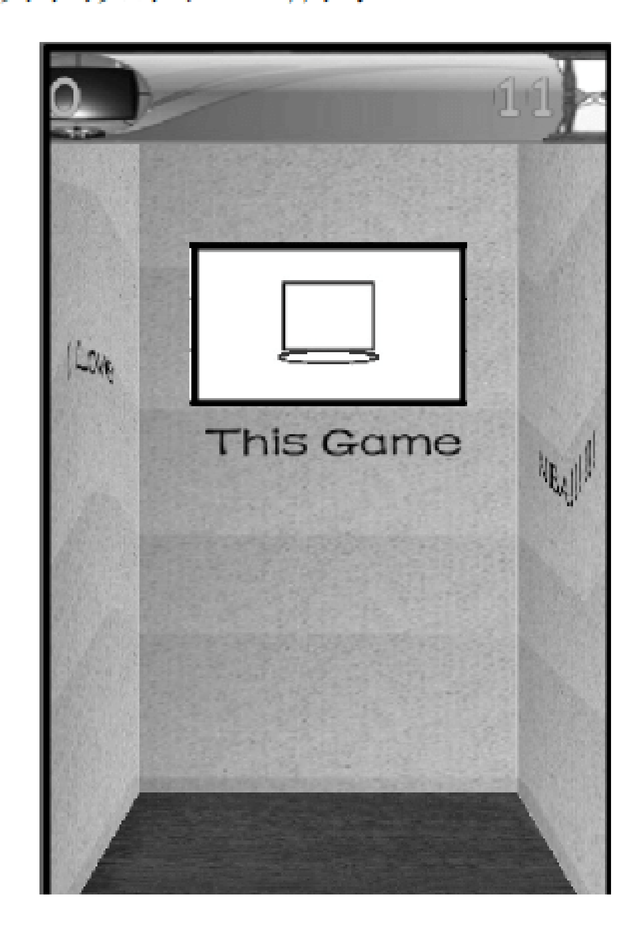


图 8-5　游戏主界面

第 9 章　象 棋 游 戏

象棋，又称中国象棋（英文现译作 Xiangqi），在中国有着悠久的历史，属于二人对抗性游戏的一种，由于用具简单、趣味性强，成为流行极为广泛的棋艺活动。中国象棋是我国正式开展的 78 个体育运动项目之一，为促进该项目在世界范围内的普及和推广，将“中国象棋”项目名称更改为“象棋”。此外，高材质的象棋也具有收藏价值，如用高档木材、玉石等材料制作的象棋。更有文人墨客为象棋谱写了诗篇，使象棋更具有一种文化色彩。本章将通过一个具体实例的实现过程，介绍开发一个 Android 象棋游戏的基本流程。

9.1　棋牌游戏介绍

知识点讲解：光盘:视频\视频讲解\第 9 章\棋牌游戏介绍.avi

棋盘游戏和牌类统称棋牌游戏。棋牌游戏从明清开始一度兴盛，涉及赌博等。现代棋牌游戏以休闲为主，在华语区影响较深的主要有扑克、斗地主、麻将、中国象棋、中国跳棋、军棋、黑白棋、五子棋、九洲游等。本节将简要介绍棋牌类游戏的发展现状和发展前景，为读者朋友们开发此类游戏做好知识铺垫。

9.1.1　棋牌游戏发展现状

随着休闲网络游戏市场竞争的升级，尤其是网络棋牌游戏的巨大市场前景的吸引，许多公司都加入到棋牌游戏的竞争之列，由于全国流行的棋牌游戏市场（牌骨类、棋类、牌类、休闲类）已经基本被几个大的游戏厂商（联众、腾讯、盛大边锋、茶苑、易发棋牌）所占据，加之棋牌游戏玩家忠诚度非常高的原因，继续介入这块市场已经没有任何意义。

由于中国不同的省份都有自己独特的文化特性，各个省份也都有自己区域内流行的棋牌游戏或者其他省份流行的棋牌游戏，把这种棋牌游戏定义为“地方棋牌游戏”，这块市场现在正处于高速成长阶段，但是在全国没有特别有影响力的品牌，其中成绩较为突出的是黄金岛、同城游、游戏茶苑等相对较大的占据几个省份市场的游戏厂商，以及其他一些非常小的游戏厂商，这块市场的竞争还没有到寡头竞争的阶段，也是由于地方游戏的特性决定的。一般来讲每一个地方游戏都是独立的，每个独立的游戏所面对的目标消费者都是不同的，所以对于企业的营销推广工作要求也比较高。

目前，地方棋牌游戏成功的模式不多，但是有一点可肯定，以后地方棋牌游戏市场的划分是以 “城市”为单位的。从市场竞争层面来看，以省份为单位的划分方法已经不能满足现在的竞争要求，不同城市的不同用户要求不能完全满足。从宏观环境来看，中国的城市化进程的加快，也需要区域性的娱乐。地方棋牌市场的划分主要是以“城市”为单位，这和中国城市化进程加快、中小城市经济发展加速是同步的，另外，地方经济的发展带动地方区域性媒体以及区域性娱乐成为可能。

9.1.2　经典游戏介绍

（1）斗地主

基础类扑克游戏，玩法简单，娱乐性强，老少皆宜。该游戏由 3 个人玩，用一副牌，共 54 张，每局牌

有一个玩家是“地主”，地主为一方，其余两玩家是“农民”，为另一方，双方对战，先出完牌的玩家所代表的一方获胜。因为每一局“地主”“农民”都会有变化，所以对抗性和配合性都很强。斗地主包括普通斗地主、特色 CT 斗地主和超级斗地主，在游戏里加入了特色任务，玩家在游戏中完成规定的任务，还有额外的奖励奉送。完成任务靠运气和技术，其乐无穷。

（2）麻将类

麻将起源于中国，属皇家和王公贵族的游戏，其历史可追溯到三四千年前。麻将的游戏人数为 4 人，分别为东、南、西、北，其中一家为庄家，其余为旁家。每人手里 13 张牌，通过吃牌、碰牌、杠牌等方式，使手牌按照相关规定的牌型条件和牌，先和牌者胜出。CT 麻将增强了麻将的娱乐性和趣味性，更有哈尔滨麻将和上海麻将等地方特色麻将。

（3）扑克类

扑克的起源众说纷纭，但它却是流行于全世界的一种娱乐游戏。玩法多种多样、精彩刺激。至尊五张，不仅需要技巧，更靠运气，可谓是现实人生的缩影。 德州扑克，易学难精，被称为是“学一时，精一世”的经典扑克游戏。 十三支，在理牌的过程中不但充满乐趣，也是对玩家理牌技术和实力的考验。更有智勇三张、角斗士、升级和锄大地，缤纷游戏，不容错过。

（4）象棋类

“运筹帷幄之中，决胜千里之外”的中国象棋，是棋艺的比拼，更体现把握棋局的能力。中国象棋给玩家一个更加公平的对弈空间。

（5）休闲类

休闲类游戏多为骰子游戏，技巧性大于运气型，对游戏者诸如观察、计算、分析、半段、反应、承受和伪装能力等综合素质均要求极高。“翻翻看”则是休闲益智游戏，考验玩家的记忆能力，简单有趣，放松娱乐。

9.2　规　划　项　目

知识点讲解：光盘:视频\视频讲解\第 9 章\规划项目.avi

在确定项目之后，开始规划此项目。项目规划的意义重大，笔者原来写程序，总是在看到功能后就马上投入到代码编写工作中，编写一个个函数去实现一个个功能。但是在后期调试时，总是会出现这样或那样的错误，需要返回重新修改。以前都是小项目，修改的工作量也不是很大，但是如果在大型项目中，几千行代码的修改是一件很恐怖的事情。所以反复强调提前规划的重要性。

9.2.1　规划流程

在项目确立以后，下一步要进行的就是游戏的大纲策划工作。

（1）大纲策划的进行

游戏大纲关系到游戏的整体面貌，当大纲策划案定稿以后，没有特殊情况，是不允许进行更改的。程序和美术工作人员将按照策划所构思的游戏形式来架构整个游戏，因此，在制定策划案时一定要做到慎重和尽量考虑成熟。

（2）正式制作

当游戏大纲策划案完成并讨论通过后，就开始进行制作。在这一阶段，策划的主要任务是在大纲的基础上对游戏的所有细节进行完善，将游戏大纲逐步填充为完整的游戏策划案。根据不同的游戏种类，所要进行细化的部分也不尽相同。

在正式制作的过程中，策划、程序、美工人员进行及时和经常性的交流，了解工作进展以及是否有难以克服的困难，并且根据现实情况有目的地变更工作计划或设计思想。三方面的配合在游戏正式制作过程中是最重要的。

（3）配音、配乐

在程序和美工快要结束时，就要进行配音和配乐的工作了。虽然音乐和音效是游戏的重要组成部分，能够起到很好的烘托游戏气氛的作用，但是限于 J2ME 游戏的开发成本和设置的处理能力，这部分已经被弱化到可有可无的地步了。但仍应选择与游戏风格能很好配合的音乐当作游戏背景音乐，这个工作交给策划比较合适。

（4）检测、调试

游戏刚制作完成，肯定在程序上会有很多错误，严重情况下会导致游戏完全没有办法进行下去。同样，策划的设计也会有不完善的地方，主要在游戏的参数部分。参数部分的不合理，会影响游戏的可玩性。此时测试人员需检测程序上的漏洞，通过试玩，调整游戏的各个部分参数使之基本平衡。

9.2.2　准备工作

在进行游戏开发之前，需要准备好游戏中用到的图片素材和配音文件。其中用到的图片素材文件保存在 res/drawable-mdpi 目录下，如图 9-2 所示。

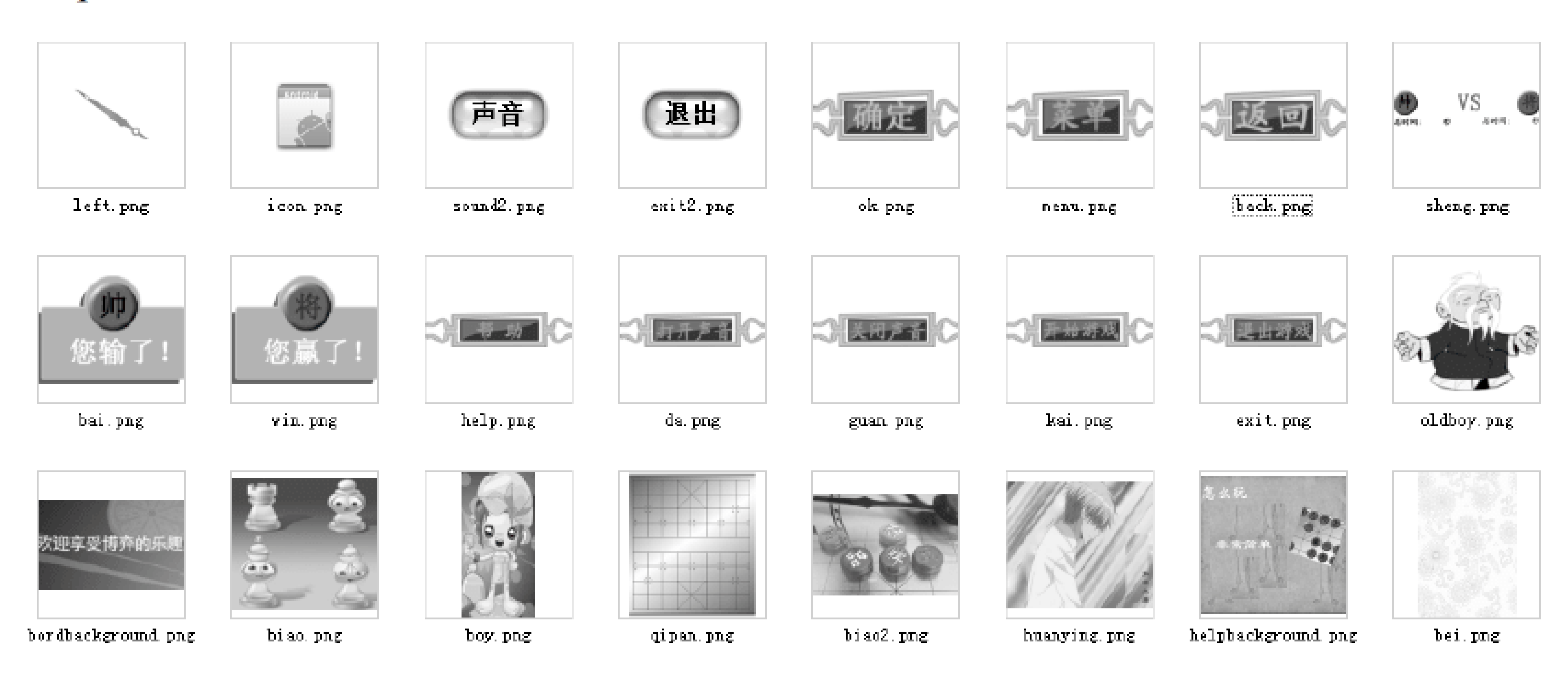


图 9-1　图片素材

用到的配音文件保存在 res/raw 目录下，如图 9-2 所示。

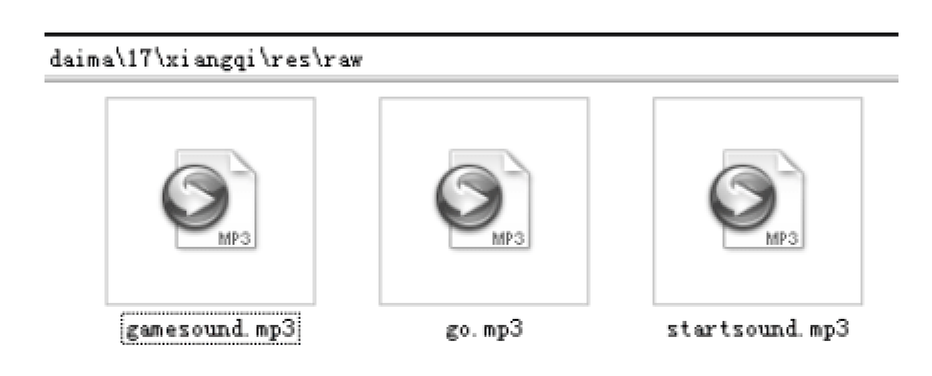
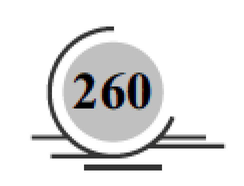


图 9-2　配音文件

9.3　项 目 架 构

知识点讲解：光盘:视频\视频讲解\第 9 章\项目架构.avi

在本节内容中，将对整个项目进行总体架构分析，并对项目中的各个类及其结构进行一一介绍。



9.3.1　总体架构

本项目的总体架构如图 9-3 所示。

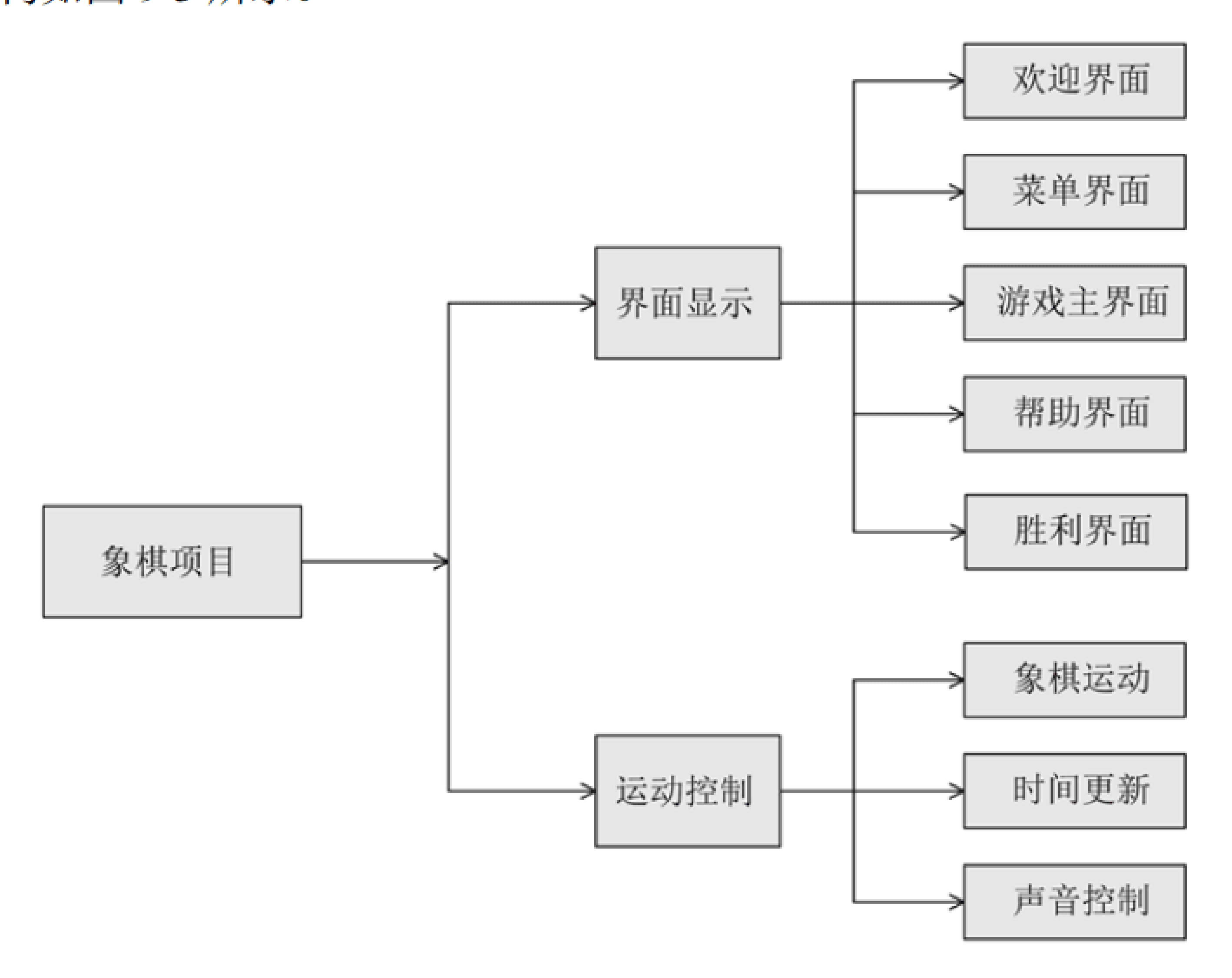


图 9-3　总体架构图

9.3.2　规划类

类是面向对象的核心，在本游戏项目中，需要编写各个类来实现具体的功能。下面将简单介绍各个类的具体功能。

1. 公有类

Activity 的实现类是 XIActivity，此类是继承于 Activity 的，是整个游戏项目的控制器，也是整个项目的入口。

2. 辅助界面类

本游戏项目中和辅助界面相关的类如下所示。

（1）欢迎界面类 Huanying

进入游戏后首先显示一个欢迎界面，此类实现欢迎界面的绘制工作。

（2）欢迎界面动画类 HuanyingThread

HuanyingThread 类是为欢迎界面类 Huanying 服务的，能够使欢迎界面以动画的效果展示在用户面前。

（3）帮助界面类 Help

Help 类用于显示本游戏项目的帮助界面。

（4）菜单界面类 Caidan

Caidan 类用于实现菜单界面，既能绘制菜单界面，也能监听菜单界面的屏幕，以便进入对应的其他界面。

3. 游戏界面类

本游戏项目中和游戏界面相关的类如下所示。

（1）游戏界面类 Game

Game 类是本游戏项目中最重要的类，用于绘制游戏过程中的所有信息，例如棋盘、棋子、按钮和胜利/失败菜单等。

（2）游戏规则类 GuiZe

GuiZe 类用于设置象棋游戏的规则，其中设置了所有棋子的走棋规则和当前棋子可走的方位。

（3）思考时间类 ShijianThread

ShijianThread 类用于计算玩家的思考时间，游戏规定思考不能超时。

（4）走法类 Move

Move 类设置了棋子的走法，包括棋子名、出发位置和终点位置。

9.4 具体编码

知识点讲解：光盘:视频\视频讲解\第 9 章\具体编码.avi

经过前面的讲解，整个项目的前期工作结束。从本节的内容开始，将进入具体编码阶段。希望读者仔细品味本节的内容，真正步入 Android 开发高手的行列。

9.4.1 实现控制类

编写文件 XIActivity.java，在此文件中定义了类 XIActivity，这是本实例游戏控制器类，功能是在合适时初始化相应的用户界面，根据其他界面的要求切换到需要的界面。文件 XIActivity.java 的主要代码如下所示。

```
public class XIActivity extends Activity {
      boolean isSound = true;                    //是否有声音
      MediaPlayer kaisheng;                      //开始的音乐
      MediaPlayer yousheng;                      //游戏时的声音

      Handler myHandler = new Handler(){//更新 UI 线程控件
           public void handleMessage(Message msg) {
             if(msg.what == 1){                  //huanyingView 或 Help 或 Game 传来的消息
                  initMenuView();                //来到菜单界面
             }
             else if(msg.what == 2){             //MenuView 传来的消息，切换到 Game
                  initGameView();                //初始游戏界面
             }
             else if(msg.what == 3){
                  initHelpView();                //初始化帮助界面
             }
           }
      };
     public void onCreate(Bundle savedInstanceState) {//重写的 onCreate()
           super.onCreate(savedInstanceState);
             //全屏
```

```
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN ,
                    WindowManager.LayoutParams.FLAG_FULLSCREEN);
        kaisheng= MediaPlayer.create(this, R.raw.kaisheng);      //欢迎音
        kaisheng.setLooping(true);                               //循环播放游戏声音
        yousheng = MediaPlayer.create(this, R.raw.yousheng);     //游戏中的背景声音
        yousheng.setLooping(true);                               //循环播放游戏声音
       this.initHuanying();                                      //欢迎界面初始化
    }
    public void initHuanying(){//欢迎界面初始化
     this.setContentView(new huanying(this,this));               //来到欢迎界面
     if(isSound){
        kaisheng.start();                                        //播放声音
        }
    }

    public void initGameView(){//游戏界面初始化
     this.setContentView(new Game(this,this));                   //来到游戏界面
    }

    public void initMenuView(){//菜单界面初始化
     if(kaisheng != null){
        kaisheng.stop();                                         //停止播放声音
        kaisheng = null;
     }
     if(this.isSound){
        yousheng.start();                                        //播放声音
     }
     this.setContentView(new caidan(this,this));                 //切换 View
    }
    public void initHelpView(){//帮助界面初始化
     this.setContentView(new Help(this,this));                   //来到帮助界面
```

9.4.2　欢迎界面类

编写文件 Huanying.java，在此文件中定义了类 Huanying，此类是一个辅助界面类，是刚进入游戏系统后显示的欢迎界面框架。文件 Huanying.java 的主要代码如下所示。

```
public class huanying extends SurfaceView implements SurfaceHolder.Callback {
    XIActivity activity;
    private TutorialThread thread;          //刷帧线程
    private huanyingThread moveThread;      //移动物件线程
    Bitmap welcomebackage;                  //大背景
    Bitmap biao;
    Bitmap boy;                             //小孩图
    Bitmap oldboy;                          //老头图
    Bitmap bordbackground;                  //文字背景
    Bitmap biao2;
    Bitmap menu;                            //菜单按钮

    int biaoX = -120;                       //移动图片坐标初始化
```

```
int boyX = -100;
int oldboyX = -120;
int biao2X = 320;

int bordbackgroundY = -100;                        //背景 y 坐标
int menuY = 520;                                   //菜单 y 坐标
public huanying(Context context,XIActivity activity) {
       super(context);
       this.activity = activity;                   //获取 Activity 引用
     getHolder().addCallback(this);
     this.thread = new TutorialThread(getHolder(), this);   //刷帧线程初始化
     this.moveThread = new huanyingThread(this);            //移动图片线程初始化
     initBitmap();//初始化全部图片
}
public void initBitmap(){//初始化全部图片方法
       welcomebackage = BitmapFactory.decodeResource(getResources(), R.drawable.huanying);
       biao = BitmapFactory.decodeResource(getResources(), R.drawable.biao);
       boy = BitmapFactory.decodeResource(getResources(), R.drawable.boy);
       oldboy = BitmapFactory.decodeResource(getResources(), R.drawable.oldboy);
       bordbackground = BitmapFactory.decodeResource(getResources(), R.drawable.bordbackground);
       biao2 = BitmapFactory.decodeResource(getResources(), R.drawable.biao2);
       menu = BitmapFactory.decodeResource(getResources(), R.drawable.menu);
}
//绘制方法，将素材图片显示在屏幕中，括号中有素材图片的名字
public void onDraw(Canvas canvas){
       canvas.drawColor(Color.BLACK);
       canvas.drawBitmap(welcomebackage, 0, 100, null);
       canvas.drawBitmap(biao, biaoX, 110, null);
       canvas.drawBitmap(boy, boyX, 210, null);
       canvas.drawBitmap(oldboy, oldboyX, 270, null);
       canvas.drawBitmap(bordbackground, 150, bordbackgroundY, null);
       canvas.drawBitmap(biao2, biao2X, 100, null);
       canvas.drawBitmap(menu, 200, menuY, null);
}
public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
}
public void surfaceCreated(SurfaceHolder holder) {//创建启动进程
     this.thread.setFlag(true);                    //循环标志位
     this.thread.start();                          //启动线程

     this.moveThread.setFlag(true);                //循环标志位
     this.moveThread.start();                      //启动线程
}
public void surfaceDestroyed(SurfaceHolder holder) {//撤销释放进程
     boolean retry = true;
     thread.setFlag(false);                        //循环标志位
     moveThread.setFlag(false);
     while (retry) {//循环
          try {
               thread.join();                      //线程结束
               moveThread.join();
```

```
                retry = false;//停止循环
            }
            catch (InterruptedException e) {//循环到刷帧线程结束
            }
        }
    }
    @Override
    public boolean onTouchEvent(MotionEvent event) {//监听屏幕
        if(event.getAction() == MotionEvent.ACTION_DOWN){
            if(event.getX()>200 && event.getX()<200+menu.getWidth()
                    && event.getY()>355 && event.getY()<355+menu.getHeight()){//单击菜单按钮
                activity.myHandler.sendEmptyMessage(1);
            }
        }
        return super.onTouchEvent(event);
    }
    class TutorialThread extends Thread{//刷帧线程
        private int span = 100;//设置睡眠 100 毫秒数
        private SurfaceHolder surfaceHolder;//SurfaceHolder 引用
        private huanying welcomeView;//WelcomeView 引用
        private boolean flag = false;
      public TutorialThread(SurfaceHolder surfaceHolder, huanying welcomeView) {//构造器
            this.surfaceHolder = surfaceHolder;
            this.welcomeView = welcomeView;
      }
      public void setFlag(boolean flag) {//循环标记位
        this.flag = flag;
      }
        @Override
        public void run() {//重写方法
            Canvas c;//画布
            while (this.flag) {//循环
                c = null;
                try {
                    //锁定画布
                    c = this.surfaceHolder.lockCanvas(null);
                    synchronized (this.surfaceHolder) {//同步
                        welcomeView.onDraw(c);//绘制
                    }
                } finally {
                    if (c != null) {
                        //更新屏幕显示内容
                         this.surfaceHolder.unlockCanvasAndPost(c);
                    }
                }
                try{
                    Thread.sleep(span);             //睡眠时间
                }
                catch(Exception e){                 //捕获异常
                    e.printStackTrace();            //输出堆栈信息
                }
```

```
                }
            }
        }
}
```

接下来需要编写文件 HuanyingThread.java，在此文件中定义了类 HuanyingThread，此类也是一个辅助界面类，用于生成欢迎界面的动画效果。文件 HuanyingThread.java 的主要代码如下所示。

```
public class huanyingThread extends Thread{
        private boolean flag = true;                              //循环标志
        huanying welcomeView;
        public huanyingThread(huanying welcomeView){//构造器
                this.welcomeView = welcomeView;
        }
        public void setFlag(boolean flag){//设置循环标志
                this.flag = flag;
        }
        public void run(){//重写方法
        try{
                Thread.sleep(300);                                //设置睡眠 300 毫秒
        }
        catch(Exception e){//捕获异常
                e.printStackTrace();                              //输出异常信息
        }
                while(flag){
                        welcomeView.biaoX += 10;                  //动态欢迎界面
                        if(welcomeView.biaoX>0){//停止移动
                                welcomeView.biaoX = 0;
                        }
                        welcomeView.boyX += 20;                   //移动男孩图片，到合适位置后停止移动
                        if(welcomeView.boyX>70){
                                welcomeView.boyX = 70;
                        }
                        welcomeView.oldboyX += 15;                //移动老头图片，到合适位置后停止移动
                        if(welcomeView.oldboyX>0){
                                welcomeView.oldboyX = 0;
                        }
                        welcomeView.bordbackgroundY += 50;        //移动文字背景
                        if(welcomeView.bordbackgroundY>240){
                                welcomeView.bordbackgroundY = 240;
                        }
                        welcomeView.biao2X -= 30;                 //更改图片坐标
                        if(welcomeView.biao2X<150){
                                welcomeView.biao2X = 150;         //停止
                        }
                        if(welcomeView.biao2X == 150){
                                welcomeView.menuY -= 30;
                                if(welcomeView.menuY<355){
                                        welcomeView.menuY = 355;
                                }
                        }
                try{
```

```
                Thread.sleep(100);          //设置睡眠 100 毫秒
            }catch(Exception e){            //捕获异常
                e.printStackTrace();        //输出异常信息
            }
        }
    }
}
```

9.4.3　菜单界面类

编写文件 Caidan.java，在此定义了类 Caidan，功能是在欢迎界面单击“菜单”按钮时进入菜单界面。文件 Caidan.java 的主要代码如下所示。

```
public class caidan extends SurfaceView implements SurfaceHolder.Callback {
    XIActivity activity;
    private TutorialThread thread;          //刷帧线程
    Bitmap kai;                             //开始图片
    Bitmap da;                              //打开声音图片
    Bitmap guan;                            //关闭声音的图片
    Bitmap help;                            //帮助图片
    Bitmap exit;                            //退出图片
    public caidan(Context context,XIActivity activity) {
        super(context);
        this.activity = activity;
      getHolder().addCallback(this);
      this.thread = new TutorialThread(getHolder(), this);//启动刷帧线程
      initBitmap();//图片资源初始化
    }
    public void initBitmap(){//图片资源初始化
        kai = BitmapFactory.decodeResource(getResources(), R.drawable.kai);           //开始按钮
        da = BitmapFactory.decodeResource(getResources(), R.drawable.da);             //开始声音按钮
        guan = BitmapFactory.decodeResource(getResources(), R.drawable.guan);         //关闭声音按钮
        help = BitmapFactory.decodeResource(getResources(), R.drawable.help);         //帮助按钮
        exit = BitmapFactory.decodeResource(getResources(), R.drawable.exit);         //退出按钮
    }
    public void onDraw(Canvas canvas){                      //绘制方法
        canvas.drawColor(Color.BLACK);                      //清屏
        canvas.drawBitmap(kai, 50, 50, null);               //绘制图片
        if(activity.isSound){//开音乐时，关闭声音图片
            canvas.drawBitmap(guan, 50, 150, null);         //关闭声音
        }else{//绘制打开声音图片
            canvas.drawBitmap(da, 50, 150, null);           //开始声音
        }
        canvas.drawBitmap(help, 50, 250, null);             //帮助按钮
        canvas.drawBitmap(exit, 50, 350, null);             //退出按钮
    }
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
    }
    public void surfaceCreated(SurfaceHolder holder) {      //启动刷帧
      this.thread.setFlag(true);                            //循环标志
      this.thread.start();                                  //启动线程
```

```
    }
    public void surfaceDestroyed(SurfaceHolder holder) {//释放刷帧线程
        boolean retry = true;                       //循环标志
        thread.setFlag(false);                      //循环标志
        while (retry) {//循环
            try {
                thread.join();                      //线程结束
                retry = false;                      //停止循环
            }catch (InterruptedException e){ }      //循环到刷帧线程结束
        }
    }
    public boolean onTouchEvent(MotionEvent event) {//监听屏幕
        if(event.getAction() == MotionEvent.ACTION_DOWN){
            if(event.getX()>105 && event.getX()<220
                    &&event.getY()>60 && event.getY()<95){//开始游戏
                activity.myHandler.sendEmptyMessage(2);
            }else if(event.getX()>105 && event.getX()<220
                    &&event.getY()>160 && event.getY()<195){//声音按钮
                activity.isSound = !activity.isSound;//取反声音开关
                if(!activity.isSound){
                    if(activity.yousheng != null){//是否正在播放声音
                        if(activity.yousheng.isPlaying()){
                            activity.yousheng.pause();//停止播放
                        }
                    }
                }else{
                    if(activity.yousheng != null){//有声音时
                        if(!activity.yousheng.isPlaying()){//没有播放的声音
                            activity.yousheng.start();//播放
                        }
                    }
                }
            }else if(event.getX()>105 && event.getX()<220
                    &&event.getY()>260 && event.getY()<295){//帮助按钮
                activity.myHandler.sendEmptyMessage(3);
            }else if(event.getX()>105 && event.getX()<220
                    &&event.getY()>360 && event.getY()<395){//退出游戏
                System.exit(0);//退出
            }
        }
        return super.onTouchEvent(event);
    }
    class TutorialThread extends Thread{//刷帧线程
        private int span = 500;//睡眠 500 毫秒
        private SurfaceHolder surfaceHolder;
        private caidan menuView;
        private boolean flag = false;//循环标记
        public TutorialThread(SurfaceHolder surfaceHolder, caidan menuView) {
            this.surfaceHolder = surfaceHolder;
            this.menuView = menuView;
        }
```

```
        public void setFlag(boolean flag) {     //循环标记
          this.flag = flag;
        }
          public void run() {                   //重写方法
                Canvas c;                       //画布
              while (this.flag) {               //循环
                    c = null;
                    try {
                        //锁定画布
                        c = this.surfaceHolder.lockCanvas(null);
                        synchronized (this.surfaceHolder) {//同步锁
                            menuView.onDraw(c);//绘制方法
                        }
                    } finally {
                        if (c != null) {
                            //更新屏幕内容
                             this.surfaceHolder.unlockCanvasAndPost(c);
                        }
                    }
                    try{
                        Thread.sleep(span);//睡眠时间，单位毫秒
                    }catch(Exception e){//捕获异常
                        e.printStackTrace();//输出印异常堆栈信息
                    }
              }
          }
    }
```

9.4.4　游戏帮助类

编写文件 Help.java，在此定义了类 Help，这也是一个辅助界面类，功能是显示游戏系统的使用方法。文件 Help.java 的主要代码如下所示。

```
public class Help extends SurfaceView implements SurfaceHolder.Callback {
    XIActivity activity;
    private TutorialThread thread;//刷帧线程
    Bitmap back;//返回按钮
    Bitmap bei;//背景图
    public Help(Context context,XIActivity activity) {
        super(context);
        this.activity = activity;//得到 Activity 引用
      getHolder().addCallback(this);
      this.thread = new TutorialThread(getHolder(), this);//重绘线程初始化
      initBitmap();//图片资源初始化
    }
    public void initBitmap(){//初始化所用到的图片
        back = BitmapFactory.decodeResource(getResources(), R.drawable.back);//返回按钮
        bei = BitmapFactory.decodeResource(
                        getResources(),
                        R.drawable.bei);//背景图片初始化
    }
```

```
public void onDraw(Canvas canvas){                        //绘制方法
        canvas.drawBitmap(bei, 0, 90, new Paint());       //绘制背景图
        canvas.drawBitmap(back, 200, 370, new Paint());   //绘制按钮
}
public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) { }
public void surfaceCreated(SurfaceHolder holder) {        //创建时启动刷帧线程
      this.thread.setFlag(true);                          //循环标志
      this.thread.start();                                //刷帧线程
}
public void surfaceDestroyed(SurfaceHolder holder) {//停止刷帧线程
      boolean retry = true;                               //循环标志
      thread.setFlag(false);                              //循环标志位
      while (retry) {
            try {
                  thread.join();                          //线程结束
                  retry = false;                          //停止循环
            }catch (InterruptedException e){ }            //循环到刷帧线程结束
      }
}
public boolean onTouchEvent(MotionEvent event) {          //监听屏幕
        if(event.getAction() == MotionEvent.ACTION_DOWN){
              if(event.getX()>200 && event.getX()<200+back.getWidth()
                          && event.getY()>370 && event.getY()<370+back.getHeight()){
                    activity.myHandler.sendEmptyMessage(1);
              }
        }
        return super.onTouchEvent(event);
}
class TutorialThread extends Thread{                      //刷帧线程
        private int span = 1000;                          //睡眠 1000 毫秒数
        private SurfaceHolder surfaceHolder;
        private Help helpView;                            //引用父类
        private boolean flag = false;                     //循环标记
      public TutorialThread(SurfaceHolder surfaceHolder, Help helpView) {
            this.surfaceHolder = surfaceHolder;
            this.helpView = helpView;
      }
      public void setFlag(boolean flag) {                 //循环标记
        this.flag = flag;
      }
        public void run() {                               //重写方法
              Canvas c;                                   //画布
            while (this.flag) {                           //循环
                  c = null;
                  try {
                        c = this.surfaceHolder.lockCanvas(null);
                        synchronized (this.surfaceHolder) {   //同步
                              helpView.onDraw(c);             //绘制方法
                        }
                  } finally {
                        if (c != null) {                  //更新屏幕显示
```

```
                        this.surfaceHolder.unlockCanvasAndPost(c);
                    }
                }
                try{
                    Thread.sleep(span);            //设置睡眠时间，单位毫秒
                }catch(Exception e){               //捕获异常
                    e.printStackTrace();           //输出异常堆栈信息
                }
            }
        }
    }
}
```

9.4.5　游戏界面框架类

编写文件 Game.java，此文件和前面介绍的界面辅助类不一样，在此文件中定义的 Game 类是一个核心类，功能是实现游戏界面框架。文件 Game.java 的实现流程如下所示。

（1）定义继承于 SurfaceView 的类 Game，然后定义了类中需要的成员变量。上述功能的实现代码如下所示。

```
package com.xiangqi;

import com.xiangqi.R;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.media.MediaPlayer;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class Game extends SurfaceView implements SurfaceHolder.Callback{
    private TutorialThread thread;      //刷帧线程
    shijianThread timeThread ;
    XIActivity activity;
    Bitmap qiPan;                       //棋盘
    Bitmap qibei;                       //棋子背景图
    Bitmap win;                         //胜利图
    Bitmap bai;                         //失败图
    Bitmap ok;                          //确定按钮
    Bitmap sheng;                       //黑方红方胜利的图
    Bitmap right;                       //向右指针
    Bitmap left;                        //向左指针
    Bitmap current;                     //“当前”文字
    Bitmap exit2;                       //退出图
    Bitmap sound2;                      //声音图
```

```
Bitmap sound3;                          //是否播放声音
Bitmap time;                            //冒号
Bitmap redtime;                         //红冒号
Bitmap background;                      //背景图
MediaPlayer go;                         //下棋声
Paint paint;                            //画笔
boolean caiPan = true;                  //玩家走棋
boolean de = false;                     //是否选中棋子
int selectqizi = 0;                     //现在选中的棋子

int startI, startJ;                     //当前棋子开始位置
int endI, endJ;                         //当前棋子目标位置
Bitmap[ ] heiZi = new Bitmap[7];        //黑子图片数组
Bitmap[ ] hongZi = new Bitmap[7];       //红子图片数组
Bitmap[ ] number = new Bitmap[10];      //数字图片数组显示时间
Bitmap[ ] redNumber = new Bitmap[10];   //红数字图片显示时间

GuiZe guiZe;                            //规则类

int status = 0;                         //游戏状态：0 游戏中，1 胜利，2 失败
int heiTime = 0;                        //黑方思考时间
int hongTime = 0;                       //红方思考时间

int[ ][ ] qizi = new int[ ][ ]{//棋盘
    {2,3,6,5,1,5,6,3,2},
    {0,0,0,0,0,0,0,0,0},
    {0,4,0,0,0,0,0,4,0},
    {7,0,7,0,7,0,7,0,7},
    {0,0,0,0,0,0,0,0,0},

    {0,0,0,0,0,0,0,0,0},
    {14,0,14,0,14,0,14,0,14},
    {0,11,0,0,0,0,0,11,0},
    {0,0,0,0,0,0,0,0,0},
    {9,10,13,12,8,12,13,10,9},
};
```

（2）分别定义系统中的构造器和对应构造方法，上述功能的实现代码如下所示。

```
public Game(Context context,XIActivity activity) {//构造器
    super(context);
    this.activity = activity;                               //得到 Activity 的引用
    getHolder().addCallback(this);
    go  = MediaPlayer.create(this.getContext(), R.raw.go);  //下棋声音
    this.thread = new TutorialThread(getHolder(), this);    //刷帧线程初始化
    this.timeThread = new shijianThread(this);              //初始化思考时间的线程
    init();//资源初始化
    guiZe = new GuiZe();                                    //规则类初始化
}

public void init(){//初始化
    paint = new Paint();                                    //画笔初始化
    qiPan = BitmapFactory.decodeResource(getResources(), R.drawable.qipan);//棋盘图
```

```
qibei = BitmapFactory.decodeResource(getResources(), R.drawable.qizi);        //棋子背景
win = BitmapFactory.decodeResource(getResources(), R.drawable.win);           //胜利图
bai = BitmapFactory.decodeResource(getResources(), R.drawable.bai);           //失败图
ok = BitmapFactory.decodeResource(getResources(), R.drawable.ok);             //确定按钮图
sheng = BitmapFactory.decodeResource(getResources(), R.drawable.sheng); //胜利图
right = BitmapFactory.decodeResource(getResources(), R.drawable.right);       //向右指针
left = BitmapFactory.decodeResource(getResources(), R.drawable.left);         //向左指针
current = BitmapFactory.decodeResource(getResources(), R.drawable.current);
exit2 = BitmapFactory.decodeResource(getResources(), R.drawable.exit2);       //退出图
sound2 = BitmapFactory.decodeResource(getResources(), R.drawable.sound2);//声音图
time = BitmapFactory.decodeResource(getResources(), R.drawable.time);         //黑冒号
redtime = BitmapFactory.decodeResource(getResources(), R.drawable.redtime);//红冒号
sound3 = BitmapFactory.decodeResource(getResources(), R.drawable.sound3);

heiZi[0] = BitmapFactory.decodeResource(getResources(), R.drawable.heishuai);    //黑帅
heiZi[1] = BitmapFactory.decodeResource(getResources(), R.drawable.heiju);       //黑车
heiZi[2] = BitmapFactory.decodeResource(getResources(), R.drawable.heima);       //黑马
heiZi[3] = BitmapFactory.decodeResource(getResources(), R.drawable.heipao);      //黑炮
heiZi[4] = BitmapFactory.decodeResource(getResources(), R.drawable.heishi);      //黑士
heiZi[5] = BitmapFactory.decodeResource(getResources(), R.drawable.heixiang);    //黑象
heiZi[6] = BitmapFactory.decodeResource(getResources(), R.drawable.heibing);     //黑兵

hongZi[0] = BitmapFactory.decodeResource(getResources(), R.drawable.hongjiang);     //红将
hongZi[1] = BitmapFactory.decodeResource(getResources(), R.drawable.hongju);        //红车
hongZi[2] = BitmapFactory.decodeResource(getResources(), R.drawable.hongma);        //红马
hongZi[3] = BitmapFactory.decodeResource(getResources(), R.drawable.hongpao);       //红炮
hongZi[4] = BitmapFactory.decodeResource(getResources(), R.drawable.hongshi);       //红士
hongZi[5] = BitmapFactory.decodeResource(getResources(), R.drawable.hongxiang);     //红相
hongZi[6] = BitmapFactory.decodeResource(getResources(), R.drawable.hongzu);        //红卒

number[0] = BitmapFactory.decodeResource(getResources(), R.drawable.number0);//黑色数字 0
number[1] = BitmapFactory.decodeResource(getResources(), R.drawable.number1);//黑色数字 1
number[2] = BitmapFactory.decodeResource(getResources(), R.drawable.number2);//黑色数字 2
number[3] = BitmapFactory.decodeResource(getResources(), R.drawable.number3);//黑色数字 3
number[4] = BitmapFactory.decodeResource(getResources(), R.drawable.number4);//黑色数字 4
number[5] = BitmapFactory.decodeResource(getResources(), R.drawable.number5);//黑色数字 5
number[6] = BitmapFactory.decodeResource(getResources(), R.drawable.number6);//黑色数字 6
number[7] = BitmapFactory.decodeResource(getResources(), R.drawable.number7);//黑色数字 7
number[8] = BitmapFactory.decodeResource(getResources(), R.drawable.number8);//黑色数字 8
number[9] = BitmapFactory.decodeResource(getResources(), R.drawable.number9);//黑色数字 9

redNumber[0] = BitmapFactory.decodeResource(getResources(), R.drawable.rednumber0);//红色数字 0
redNumber[1] = BitmapFactory.decodeResource(getResources(), R.drawable.rednumber1);//红色数字 1
redNumber[2] = BitmapFactory.decodeResource(getResources(), R.drawable.rednumber2);//红色数字 2
redNumber[3] = BitmapFactory.decodeResource(getResources(), R.drawable.rednumber3);//红色数字 3
redNumber[4] = BitmapFactory.decodeResource(getResources(), R.drawable.rednumber4);//红色数字 4
redNumber[5] = BitmapFactory.decodeResource(getResources(), R.drawable.rednumber5);//红色数字 5
redNumber[6] = BitmapFactory.decodeResource(getResources(), R.drawable.rednumber6);//红色数字 6
redNumber[7] = BitmapFactory.decodeResource(getResources(), R.drawable.rednumber7);//红色数字 7
redNumber[8] = BitmapFactory.decodeResource(getResources(), R.drawable.rednumber8);//红色数字 8
redNumber[9] = BitmapFactory.decodeResource(getResources(), R.drawable.rednumber9);//红色数字 9
```

```
            background = BitmapFactory.decodeResource(getResources(), R.drawable.bei);
    }
```

（3）定义绘制方法 onDraw()，该方法是自己定义的并非重写的，只会根据数据绘制屏幕。上述功能的主要实现代码如下所示。

```
    public void onDraw(Canvas canvas){//根据数据绘制屏幕方法
        canvas.drawColor(Color.WHITE);
        canvas.drawBitmap(background, 0,0, null);//清背景
        canvas.drawBitmap(qiPan, 10, 10, null);//绘制棋盘
        for(int i=0; i<qizi.length; i++){
            for(int j=0; j<qizi[i].length; j++){//绘制棋子
                if(qizi[i][j] != 0){
                    canvas.drawBitmap(qibei, 9+j*34, 10+i*35, null);//绘制棋子背景
                    if(qizi[i][j] == 1){//黑帅时
                        canvas.drawBitmap(heiZi[0], 12+j*34, 13+i*35, paint);
                    }
                    else if(qizi[i][j] == 2){//黑车时
                        canvas.drawBitmap(heiZi[1], 12+j*34, 13+i*35, paint);
                    }
                    else if(qizi[i][j] == 3){//黑马时
                        canvas.drawBitmap(heiZi[2], 12+j*34, 13+i*35, paint);
                    }
                    else if(qizi[i][j] == 4){//黑炮时
                        canvas.drawBitmap(heiZi[3], 12+j*34, 13+i*35, paint);
                    }
                    else if(qizi[i][j] == 5){//黑士时
                        canvas.drawBitmap(heiZi[4], 12+j*34, 13+i*35, paint);
                    }
                    else if(qizi[i][j] == 6){//黑象时
                        canvas.drawBitmap(heiZi[5], 12+j*34, 13+i*35, paint);
                    }
                    else if(qizi[i][j] == 7){//黑兵时
                        canvas.drawBitmap(heiZi[6], 12+j*34, 13+i*35, paint);
                    }
                    else if(qizi[i][j] == 8){//红将时
                        canvas.drawBitmap(hongZi[0], 12+j*34, 13+i*35, paint);
                    }
                    else if(qizi[i][j] == 9){//红车时
                        canvas.drawBitmap(hongZi[1], 12+j*34, 13+i*35, paint);
                    }
                    else if(qizi[i][j] == 10){//红马时
                        canvas.drawBitmap(hongZi[2], 12+j*34, 13+i*35, paint);
                    }
                    else if(qizi[i][j] == 11){//红炮时
                        canvas.drawBitmap(hongZi[3], 12+j*34, 13+i*35, paint);
                    }
                    else if(qizi[i][j] == 12){//红士时
                        canvas.drawBitmap(hongZi[4], 12+j*34, 13+i*35, paint);
                    }
                    else if(qizi[i][j] == 13){//红相时
                        canvas.drawBitmap(hongZi[5], 12+j*34, 13+i*35, paint);
```

```
                        }
                        else if(qizi[i][j] == 14){//红卒时
                            canvas.drawBitmap(hongZi[6], 12+j*34, 13+i*35, paint);
                        }
                    }
                }
            }
            canvas.drawBitmap(sheng, 10, 360, paint);//绘制声音背景图
            //绘制黑方的时间
            canvas.drawBitmap(time, 81, 411, paint);//冒号
            int temp = this.heiTime/60;//时间换算
            String timeStr = temp+"";//转为字符串
            if(timeStr.length()<2){//不足两位时前面填 0
                timeStr = "0" + timeStr;
            }
        for(int i=0;i<2;i++){//循环绘制时间
            int tempScore=timeStr.charAt(i)-'0';
            canvas.drawBitmap(number[tempScore], 65+i*7, 412, paint);
        }
        //绘制分钟
        temp = this.heiTime%60;
            timeStr = temp+"";//转成字符串
            if(timeStr.length()<2){
                timeStr = "0" + timeStr;//长度小于 2 时在前面添加一个 0
            }
        for(int i=0;i<2;i++){//循环
            int tempScore=timeStr.charAt(i)-'0';
            canvas.drawBitmap(number[tempScore], 85+i*7, 412, paint);//绘制
        }
        //绘制红方时间
            canvas.drawBitmap(this.redtime, 262, 410, paint);//红冒号
            int temp2 = this.hongTime/60;//换算时间
            String timeStr2 = temp2+"";//转成字符串
            if(timeStr2.length()<2){//不足两位时前面填 0
                timeStr2 = "0" + timeStr2;
            }
        for(int i=0;i<2;i++){//循环绘制时间
            int tempScore=timeStr2.charAt(i)-'0';
            canvas.drawBitmap(redNumber[tempScore], 247+i*7, 411, paint);//绘制
        }
        //绘制分钟
        temp2 = this.hongTime%60;//当前秒数
            timeStr2 = temp2+"";//转换成字符串
            if(timeStr2.length()<2){//不足两位时前面用 0 补
            timeStr2 = "0" + timeStr2;
            }
        for(int i=0;i<2;i++){//循环绘制
            int tempScore=timeStr2.charAt(i)-'0';
            canvas.drawBitmap(redNumber[tempScore], 267+i*7, 411, paint);//绘制时间数字
        }
            if(caiPan == true){
```

```
                canvas.drawBitmap(right, 155, 420, paint);//向右指针
            }
            else{//黑方走棋，即计算机走棋时
                canvas.drawBitmap(left, 120, 420, paint);//向左指针
            }

            canvas.drawBitmap(current, 138, 445, paint);//当前文字
            canvas.drawBitmap(sound2, 10, 440, paint);//绘制声音
            if(activity.isSound){//如果正在播放声音则绘制
                canvas.drawBitmap(sound3, 80, 452, paint);
            }

            canvas.drawBitmap(exit2, 250, 440, paint);//绘制退出按钮
            if(status == 1){//胜利时
                canvas.drawBitmap(win, 85, 150, paint);//绘制胜利界面
                canvas.drawBitmap(ok, 113, 240, paint);
            }
            if(status == 2){//失败
                canvas.drawBitmap(bai, 85, 150, paint);//绘制失败界面
                canvas.drawBitmap(ok, 113, 236, paint);
            }
        }
```

（4）定义重写的屏幕监听方法 onTouchEvent()，该方法是游戏主要逻辑接口，用于接受玩家输入，会根据单击的位置和当前的游戏状态做出相应的处理，当需要切换 View 界面时，通过给 Activity 发送 Handler 消息来处理。上述功能的实现代码如下所示。

```
        public boolean onTouchEvent(MotionEvent event) {//屏幕监听重写
            if(event.getAction() == MotionEvent.ACTION_DOWN){//鼠标按下事件
                if(event.getX()>10&&event.getX()<10+sound2.getWidth()
                        && event.getY()>440 && event.getY()<440+sound2.getHeight()){//单击声音按钮
                    activity.isSound = !activity.isSound;
                    if(activity.isSound){//当播放声音时
                        if(activity.yousheng != null){
                            if(!activity.yousheng.isPlaying()){
                            activity.yousheng.start();//播放音乐
                            }
                        }
                    }
                    else{
                        if(activity.yousheng != null){
                            if(activity.yousheng.isPlaying()){
                                activity.yousheng.pause();//停止音乐
                            }
                        }
                    }
                }
                if(event.getX()>250&&event.getX()<250+exit2.getWidth()
                        && event.getY()>440 && event.getY()<440+exit2.getHeight()){//单击退出按钮
                    activity.myHandler.sendEmptyMessage(1);
                }
                if(status == 1){//胜利
```

```
        if(event.getX()>135&&event.getX()<190
                && event.getY()>249 && event.getY()<269){
            activity.myHandler.sendEmptyMessage(1);//发送消息
        }
    }
    else if(status == 2){//失败后
        if(event.getX()>135&&event.getX()<190
                && event.getY()>245 && event.getY()<265){//单击了确定按钮
            activity.myHandler.sendEmptyMessage(1);//发送消息，切换到 MenuView
        }
    }


    else if(status == 0){//游戏中时
        if(event.getX()>10&&event.getX()<310
                && event.getY()>10 && event.getY()<360){//单击位置在棋盘内
                if(caiPan == true){//如果是该玩家走棋
                    int i = -1, j = -1;
                    int[ ] pos = getPos(event);//根据坐标换算成所在的行和列
                    i = pos[0];
                    j = pos[1];
                    if(de == false){//前面没有选中的棋子
                        if(qizi[i][j] != 0){//单击的位置有棋子
                            if(qizi[i][j] > 7){//单击的是自己的棋子
                                selectqizi = qizi[i][j];//设为选中的棋子
                                de = true;//标记有选中的棋子
                                startI = i;
                                startJ = j;
                            }
                        }
                    }
                    else{//之前选中过棋子
                        if(qizi[i][j] != 0){//单击的位置有棋子
                            if(qizi[i][j] > 7){//如果是自己的棋子
                                selectqizi = qizi[i][j];//将该棋子设为选中的棋子
                                startI = i;
                                startJ = j;
                            }
                            else{//如果是对方的棋子
                                endI = i;
                                endJ = j;//保存点
                        boolean canMove = guiZe.canMove(qizi, startI, startJ, endI, endJ);
                                if(canMove){//可以移动过去
                                    caiPan = false;//不让玩家走
                            if(qizi[endI][endJ] == 1 || qizi[endI][endJ] == 8){//是“帅”或“将”
                                        this.success();//胜利
                                    }
                                    else{
                                        if(activity.isSound){
                                            go.start();//播放下棋声
                                        }
```

```
                                        qizi[endI][endJ] = qizi[startI][startJ];//移动棋子
                                        qizi[startI][startJ] = 0;
                                        startI = -1;
                                        startJ = -1;
                                        endI = -1;
                                        endJ = -1;//还原保存点
                                        de = false;//当前没有选中棋子

                                        Move cm = guiZe.searchAGoodMove(qizi);
                                        if(activity.isSound){
                                            go.start();//播放下棋声
                                        }
                                        qizi[cm.toX][cm.toY] = qizi[cm.qiX][cm.qiY];//移动棋子
                                        qizi[cm.qiX][cm.qiY] = 0;
                                        caiPan = true;
                                    }
                                }
                            }
                        }
                        else{//如果没有棋子
                            endI = i;
                            endJ = j;
                            boolean canMove = guiZe.canMove(qizi, startI, startJ, endI,
                            endJ);//查看是否可走
                            if(canMove){//如果可以移动
                                caiPan = false;//不让玩家走
                                if(activity.isSound){
                                    go.start();//播放下棋声
                                }
                                qizi[endI][endJ] = qizi[startI][startJ];//动棋子
                                qizi[startI][startJ] = 0;//置空原来位置
                                startI = -1;
                                startJ = -1;
                                endI = -1;
                                endJ = -1;//还原保存点
                                de = false;//标志为 false

                                Move cm = guiZe.searchAGoodMove(qizi);//得到走法
                                if(qizi[cm.toX][cm.toY] == 8){//吃了将
                                    status = 2;//失败
                                }
                                if(activity.isSound){
                                    go.start();
                                }
                                qizi[cm.toX][cm.toY] = qizi[cm.qiX][cm.qiY];//移动棋子
                                qizi[cm.qiX][cm.qiY] = 0;
                                caiPan = true;
                            }
                        }
                    }
                }
```

```
                    }
                }
            }
            return super.onTouchEvent(event);
        }
```

（5）定义方法 getPos()，用于将坐标换算成数组的维数；定义方法 success()表示胜利。上述功能的实现代码如下所示。

```
        public int[ ] getPos(MotionEvent e){//坐标转换成数组维数
            int[ ] pos = new int[2];
            double x = e.getX();//单击位置 x 坐标
            double y = e.getY();//单击位置 y 坐标
            if(x>10 && y>10 && x<10+qiPan.getWidth() && y<10+qiPan.getHeight()){//单击棋盘时
                pos[0] = Math.round((float)((y-21)/36));//获取所在行
                pos[1] = Math.round((float)((x-21)/35));//获取所在列
            }
            else{//单击的不是棋盘时
                pos[0] = -1;//设置位置不可用
                pos[1] = -1;
            }
            return pos;//返回坐标数组
        }

        public void success(){//胜利
            status = 1;//来到胜利状态
        }
```

9.4.6 象棋走法类

编写文件 Move.java，在此文件中定义了象棋的走法类 Move，在走法中包含了什么棋子、起始点的位置、目标点的位置以及估值时所用到的得分 score。文件 Move.java 的主要代码如下所示。

```
    public class Move {
        int ChessID;//什么棋子
        int qiX;//起始坐标
        int qiY;
        int toX;//目标坐标
        int toY;
        int score;//得分
        public Move(int ChessID, int qiX,int qiY,int toX,int toY,int score){//构造器
            this.ChessID = ChessID;//棋子类型
            this.qiX = qiX;//起始坐标
            this.qiY = qiY;
            this.toX = toX;//目标 x 坐标
            this.toY = toY;//目标 y 坐标
            this.score = score;
        }
    }
```

9.4.7 思考时间类

编写文件 ShijianThread.java，在此文件中定义了 ShijianThread 类，此类能够根据是哪一方该走子，并增加这一方的思考时间。文件 shijianThread.java 的主要代码如下所示。

```
public class shijianThread extends Thread{
	private boolean flag = true;//循环标志
	Game gameView;
	public shijianThread(Game gameView){
		this.gameView = gameView;
	}
	public void setFlag(boolean flag){//设置循环标记
		this.flag = flag;
	}
	@Override
	public void run(){//重写方法
		while(flag){//循环
			if(gameView.caiPan == false){//自动加黑方时间
			}
			else if(gameView.caiPan == true){//为红方走棋、思考
				gameView.hongTime++;//自动加红方时间
			}
			try{
				Thread.sleep(1000);//睡眠 1000 毫秒，即 1 秒钟
			}
			catch(Exception e){//捕获异常
				e.printStackTrace();//输出异常信息
			}
		}
	}
}
```

9.4.8 走法规则类

编写文件 GuiZe.java，其中定义了象棋规则类 GuiZe。象棋是有规则的，例如马走日、象走方，主要代码如下所示。

```
public class GuiZe {
	boolean isRedGo = false;//是否红方走棋
	public boolean canMove(int[ ][ ] qizi, int fromY, int fromX, int toY, int toX){
		int i = 0;
		int j = 0;
		int qiID;//起始位置的棋子类型
		int muID;//目标位置
		if(toX<0){//左边出界
			return false;
		}
		if(toX>8){//右边出界
			return false;
		}
		if(toY<0){//上边出界
```

```
            return false;
        }
        if(toY>9){//下边出界
            return false;
        }
        if(fromX==toX && fromY==toY){//目标位置与起始位置相同
            return false;
        }
        qiID = qizi[fromY][fromX];//起始棋子
        muID = qizi[toY][toX];
        if(isSameSide(qiID,muID)){//一伙的
            return false;
        }
        switch(qiID){
            case 1://黑帅
                if(toY>2||toX<3||toX>5){//出了九宫格
                    return false;
                }
                if((Math.abs(fromY-toY)+Math.abs(toX-fromX))>1){//只能走一步
                    return false;
                }
                break;
            case 5://黑士
                if(toY>2||toX<3||toX>5){//出了九宫格
                    return false;
                }
                if(Math.abs(fromY-toY) != 1 || Math.abs(toX-fromX) != 1){//走斜线
                    return false;
                }
                break;
            case 6://黑象
                if(toY>4){//不过河
                    return false;
                }
                if(Math.abs(fromX-toX) != 2 || Math.abs(fromY-toY) != 2){//相走田
                    return false;
                }
                if(qizi[(fromY+toY)/2][(fromX+toX)/2] != 0){
                    return false;
                }
                break;
            case 7://黑兵
                if(toY < fromY){//不回头
                    return false;
                }
                if(fromY<5 && fromY == toY){//过河前只能直走
                    return false;
                }
                if(toY - fromY + Math.abs(toX-fromX) > 1){//只走一步直线
                    return false;
                }
                break;
            case 8://红将
```

```
                if(toY<7||toX>5||toX<3){//出了九宫格
                    return false;
                }
                if((Math.abs(fromY-toY)+Math.abs(toX-fromX))>1){//只走一步
                    return false;
                }
                break;
            case 2://黑车
            case 9://红车
                if(fromY != toY && fromX != toX){//只走直线
                    return false;
                }
                if(fromY == toY){//走横线
                    if(fromX < toX){//向右走
                        for(i = fromX + 1; i < toX; i++){//循环
                            if(qizi[fromY][i] != 0){
                                return false;//返回 false
                            }
                        }
                    }
                    else{//向左走
                        for(i = toX + 1; i < fromX; i++){//循环
                            if(qizi[fromY][i] != 0){
                                return false;//返回 false
                            }
                        }
                    }
                }
                else{//走的是竖线
                    if(fromY < toY){//向右走
                        for(j = fromY + 1; j < toY; j++){
                            if(qizi[j][fromX] != 0)
                            return false;//返回 false
                        }
                    }
                    else{//向左走
                        for(j= toY + 1; j < fromY; j++){
                            if(qizi[j][fromX] != 0)
                            return false;//返回 false
                        }
                    }
                }
                break;
            case 10://红马
            case 3://黑马
                if(!((Math.abs(toX-fromX)==1 && Math.abs(toY-fromY)==2)
                        || (Math.abs(toX-fromX)==2 && Math.abs(toY-fromY)==1))){
                    return false;//不是日时
                }
                if(toX-fromX==2){//向右
                    i=fromX+1;//移动
                    j=fromY;
                }
```

```
            else if(fromX-toX==2){//向左
                    i=fromX-1;
                    j=fromY;
            }
            else if(toY-fromY==2){//向下
                    i=fromX;
                    j=fromY+1;
            }
            else if(fromY-toY==2){//向上
                    i=fromX;
                    j=fromY-1;
            }
            if(qizi[j][i] != 0)
                    return false;//蹩马腿
            break;
    case 11://红炮
    case 4://黑炮
            if(fromY!=toY && fromX!=toX){//走直线
                    return false;
            }
            if(qizi[toY][toX] == 0){
                    if(fromY == toY){//横线
                            if(fromX < toX){//向右走
                                    for(i = fromX + 1; i < toX; i++){
                                            if(qizi[fromY][i] != 0){
                                                    return false;
                                            }
                                    }
                            }
                            else{//向左走
                                    for(i = toX + 1; i < fromX; i++){
                                            if(qizi[fromY][i]!=0){
                                                    return false;
                                            }
                                    }
                            }
                    }
                    else{//竖线
                            if(fromY < toY){//向下走
                                    for(j = fromY + 1; j < toY; j++){
                                            if(qizi[j][fromX] != 0){
                                                    return false;//返回 false
                                            }
                                    }
                            }
                            else{//向上走
                                    for(j = toY + 1; j < fromY; j++){
                                            if(qizi[j][fromX] != 0){
                                                    return false;//返回 false
                                            }
                                    }
                            }
                    }
```

```
                }
                else{//吃子
                    int count=0;
                    if(fromY == toY){//走横线
                        if(fromX < toX){//向右
                            for(i=fromX+1;i<toX;i++){
                                if(qizi[fromY][i]!=0){
                                    count++;
                                }
                            }
                            if(count != 1){
                                return false;//返回 false
                            }
                        }
                        else{//向左走
                            for(i=toX+1;i<fromX;i++){
                                if(qizi[fromY][i] != 0){
                                    count++;
                                }
                            }
                            if(count!=1){
                                return false;//返回 false
                            }
                        }
                    }
                    else{//走竖线
                        if(fromY<toY){//向下走
                            for(j=fromY+1;j<toY;j++){
                                if(qizi[j][fromX]!=0){
                                    count++;
                                }
                            }
                            if(count!=1){
                                return false;
                            }
                        }
                        else{//向上走
                            for(j=toY+1;j<fromY;j++){
                                if(qizi[j][fromX] != 0){
                                    count++;//返回 false
                                }
                            }
                            if(count!=1){
                                return false;//返回 false
                            }
                        }
                    }
                }
                break;
            case 12://红士
                if(toY<7||toX>5||toX<3){//出了九宫格
                    return false;
                }
```

```
                if(Math.abs(fromY-toY) != 1 || Math.abs(toX-fromX) != 1){//走斜线
                    return false;
                }
                break;
            case 13://红相
                if(toY<5){//不能过河
                    return false;//返回 false
                }
                if(Math.abs(fromX-toX) != 2 || Math.abs(fromY-toY) != 2){//相走田字
                    return false;//返回 false
                }
                if(qizi[(fromY+toY)/2][(fromX+toX)/2] != 0){
                    return false;//相眼有棋子
                }
                break;
            case 14://红卒
                if(toY > fromY){//不回头
                    return false;
                }
                if(fromY > 4 && fromY == toY){
                    return false;//不让走
                }
                if(fromY - toY + Math.abs(toX - fromX) > 1){//只能走一步直线
                    return false;//返回 false 不让走
                }
                break;
            default:
                return false;
        }
        return true;
    }
```

至于计算机的走法，实现原理和上述代码类似。为了节省本书篇幅，不再进行详细的讲解。读者只需参阅本书光盘中的内容即可。到此为止，此项目的主要功能介绍完毕，执行后的初始界面效果如图 9-4 所示，游戏界面效果如图 9-5 所示。



图 9-4　初始效果

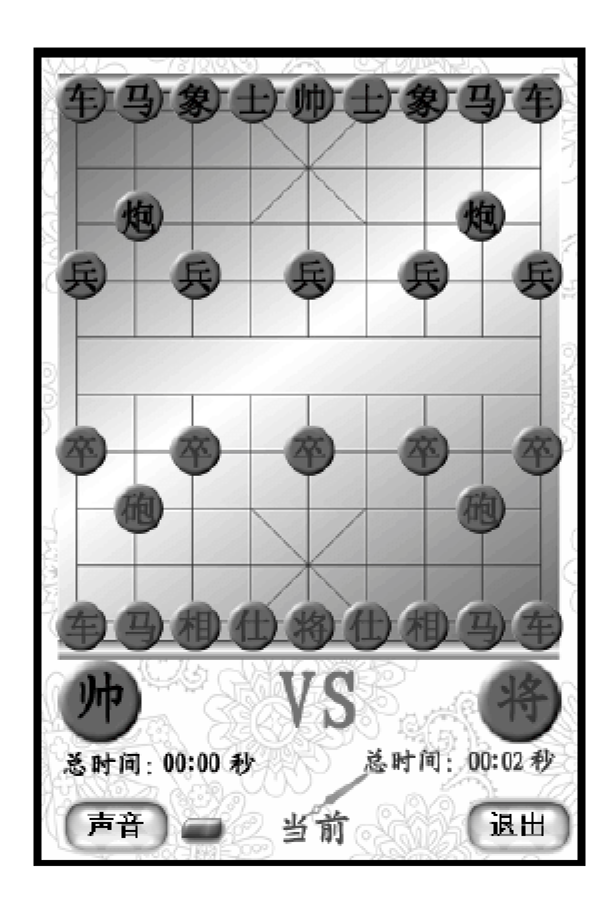


图 9-5　游戏界面效果

第 10 章　暴走轨迹计步器

暴走是指沿着指定路线徒步或驾车行走，时间不限。暴走这项运动源于美国，风靡欧美，是一种高强度又简单易行的户外运动方式，目前世界上暴走一族大约有七千万人。日渐流行的“暴走”，几乎成了时尚、健身、释放、减压的代名词。近年来，全民健身热潮的兴起为传感器应用开发提供了良好的舞台。本章通过一个综合实例的实现过程，详细讲解利用 Android 传感器技术开发暴走轨迹计步器系统的方法，为读者步入现实工作岗位打下基础。

10.1　系统功能模块介绍

知识点讲解：光盘:视频\视频讲解\第 10 章\系统功能模块介绍.avi

本章轨迹记录器的功能是，通过 Android 传感器记录当前的位置、速率、海拔、记录频率和距离等信息，并且可以将轨迹信息打包上传或分享。本章暴走轨迹记录器的构成模块结构如图 10-1 所示。

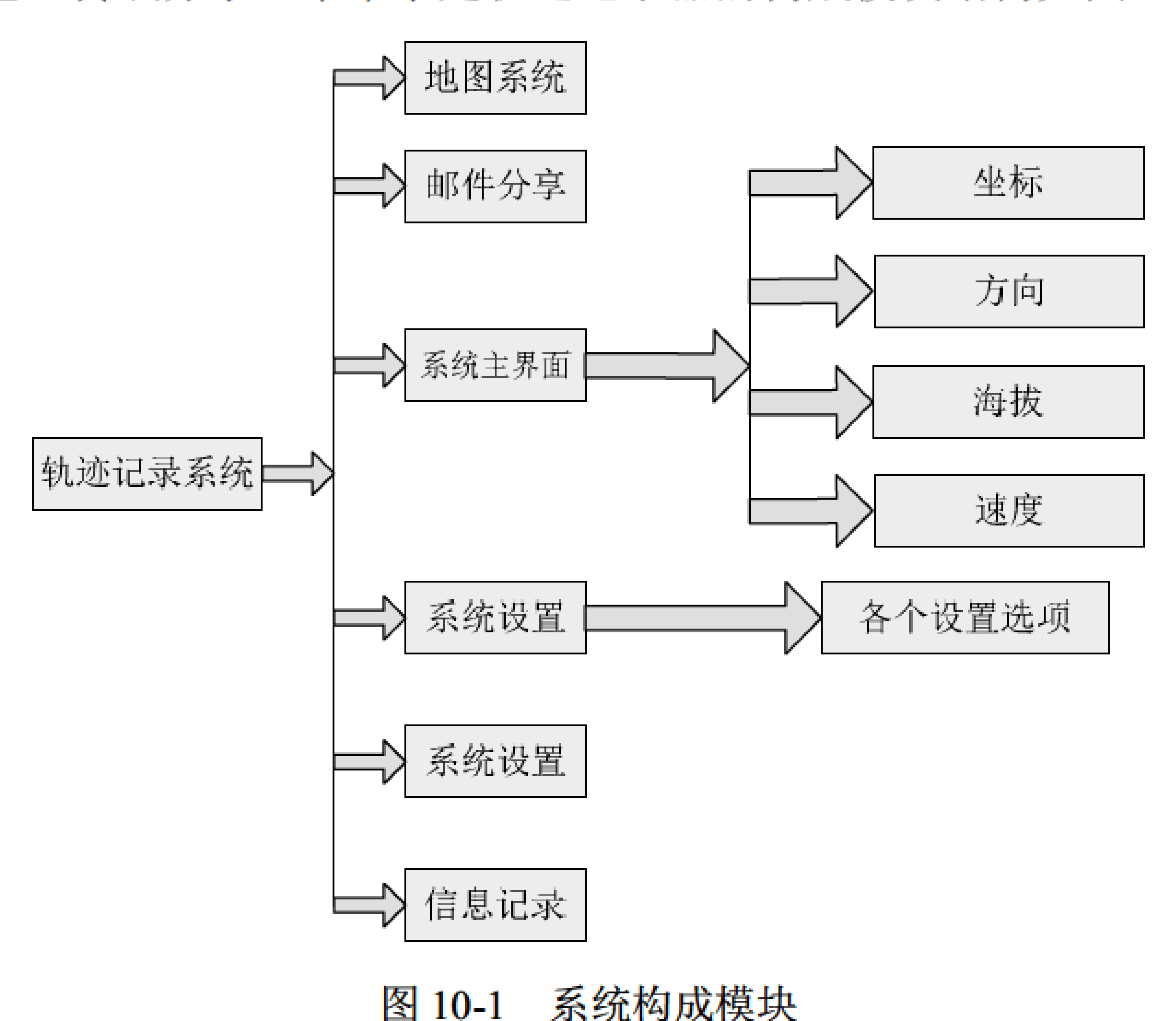


图 10-1　系统构成模块

10.2　系统主界面

知识点讲解：光盘:视频\视频讲解\第 10 章\系统主界面.avi

系统主界面是运行程序后首先呈现在用户面前的界面。在本节的内容中，将详细讲解本章暴走轨迹记录器主界面的具体实现流程。

10.2.1　布局文件

本系统主界面的布局文件是 main.xml，功能是通过文本控件显示当前的位置信息和传感器信息，具体实现代码如下所示。

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
            android:id="@+id/scroll" android:layout_width="fill_parent"
            android:layout_height="fill_parent" android:background="#000000">
    <LinearLayout
                android:layout_width="fill_parent" android:layout_height="fill_parent"
                android:orientation="vertical">
        <TextView android:id="@+id/textStatus" android:layout_width="wrap_content"
                android:layout_height="wrap_content"/>
        <TableLayout android:id="@+id/TableGPS"
                  android:layout_width="fill_parent" android:layout_height="wrap_content"
                  android:stretchColumns="1" android:background="#000000">
            <TableRow android:background="#333333" android:layout_margin="1dip">
                <TextView android:id="@+id/txtDateTimeAndProvider"
                        android:gravity="left" android:textStyle="bold" android:padding="2dip"
                        android:layout_span="2"/>
            </TableRow>
            <TableRow android:background="#333333" android:layout_margin="1dip">
                <TextView android:textStyle="bold" android:text="@string/txt_latitude"
                        android:padding="3dip" android:textSize="17sp"/>
                <TextView android:id="@+id/txtLatitude" android:gravity="left"
                        android:padding="3dip" android:textColor="#e8a317"
                        android:textStyle="bold" android:textSize="18sp"/>
            </TableRow>
            <TableRow android:background="#333333" android:layout_margin="1dip">
                <TextView android:textStyle="bold" android:text="@string/txt_longitude"
                        android:padding="3dip" android:textSize="17sp"/>
                <TextView android:id="@+id/txtLongitude" android:gravity="left"
                        android:padding="3dip" android:textColor="#e8a317"
                        android:textStyle="bold" android:textSize="18sp"/>
            </TableRow>
            <TableRow android:background="#333333" android:layout_margin="1dip">
                <TextView android:id="@+id/lblAltitude" android:textStyle="bold"
                        android:text="@string/txt_altitude" android:padding="3dip"/>
                <TextView android:id="@+id/txtAltitude" android:gravity="left"
                        android:padding="3dip"/>
            </TableRow>
            <TableRow android:background="#333333" android:layout_margin="1dip">
                <TextView android:id="@+id/lblSpeed" android:textStyle="bold"
                        android:text="@string/txt_speed" android:padding="3dip"/>
                <TextView android:id="@+id/txtSpeed" android:gravity="left"
                        android:padding="3dip"/>
            </TableRow>
            <TableRow android:background="#333333" android:layout_margin="1dip">
                <TextView android:id="@+id/lblDirection" android:textStyle="bold"
                        android:text="@string/txt_direction" android:padding="3dip"/>
                <TextView android:id="@+id/txtDirection" android:gravity="left"
```

```
                    android:padding="3dip"/>
        </TableRow>
        <TableRow android:background="#333333" android:layout_margin="1dip">
            <TextView android:id="@+id/lblSatellites" android:textStyle="bold"
                    android:text="@string/txt_satellites" android:padding="3dip"/>
            <TextView android:id="@+id/txtSatellites" android:gravity="left"
                    android:padding="3dip"/>
        </TableRow>
        <TableRow android:background="#333333" android:layout_margin="1dip">
            <TextView android:id="@+id/lblAccuracy" android:textStyle="bold"
                    android:text="@string/txt_accuracy" android:padding="3dip"/>
            <TextView android:id="@+id/txtAccuracy" android:gravity="left"
                    android:padding="3dip"/>
        </TableRow>
    </TableLayout>
    <LinearLayout android:orientation="vertical"
                android:layout_width="fill_parent" android:layout_height="wrap_content">
        <!--
            <Button android:id="@+id/buttonStart" android:layout_width="120px"
            android:layout_height="wrap_content" android:tag="Start"
            android:text="Start" /> <Button android:id="@+id/buttonStop"
            android:layout_width="120px" android:layout_height="wrap_content"
            android:tag="Stop" android:text="Stop" />
        -->
        <ToggleButton android:id="@+id/buttonOnOff"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:textOn="@string/btn_stop_logging"
                    android:textOff="@string/btn_start_logging"/>
    </LinearLayout>
    <TableLayout android:id="@+id/TableSummary"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:background="#222222">
        <TableRow android:layout_width="fill_parent"
                    android:layout_height="fill_parent">
            <TextView android:id="@+id/lblLoggingTo" android:layout_width="wrap_content"
                    android:textSize="9dip"
                    android:layout_height="fill_parent"
                    android:textStyle="italic"
                    android:paddingLeft="8dip" android:text="@string/summary_loggingto"/>
            <TextView android:id="@+id/txtLoggingTo"
                    android:layout_width="wrap_content"
                    android:paddingLeft="3dip"
                    android:textSize="9dip" android:textStyle="italic"
                    android:layout_height="fill_parent"/>
        </TableRow>
        <TableRow android:layout_width="fill_parent"
                android:layout_height="fill_parent">
            <TextView android:id="@+id/lblFrequency" android:layout_width="wrap_content"
                    android:textSize="9dip" android:layout_height="fill_parent" android:textStyle="italic"
```

```
                    android:paddingLeft="8dip" android:text="@string/summary_freq_every"/>
                <TextView android:id="@+id/txtFrequency" android:layout_width="wrap_content"
                    android:paddingLeft="3dip" android:textSize="9dip" android:textStyle="italic"
                    android:layout_height="fill_parent"/>
            </TableRow>
            <TableRow android:layout_width="fill_parent"
                    android:layout_height="fill_parent">
                <TextView android:id="@+id/lblDistance" android:layout_width="wrap_content"
                    android:textSize="9dip" android:layout_height="fill_parent" android:textStyle="italic"
                    android:paddingLeft="8dip" android:text="@string/summary_dist"/>
                <TextView android:id="@+id/txtDistance" android:layout_width="wrap_content"
                    android:paddingLeft="3dip" android:textSize="9dip" android:textStyle="italic"
                    android:layout_height="fill_parent"/>
            </TableRow>
            <TableRow android:layout_width="fill_parent"
                    android:layout_height="fill_parent">
                <TextView android:id="@+id/lblFileName" android:layout_width="wrap_content"
                    android:textSize="9dip" android:layout_height="fill_parent" android:textStyle="italic"
                    android:paddingLeft="8dip" android:text="@string/summary_current_filename"/>
                <TextView android:id="@+id/txtFileName" android:layout_width="wrap_content"
                    android:paddingLeft="3dip" android:textSize="9dip" android:textStyle="italic"
                    android:layout_height="fill_parent"/>
            </TableRow>
            <TableRow android:layout_width="fill_parent"
                    android:layout_height="fill_parent" android:id="@+id/trAutoEmail">
                <TextView android:id="@+id/lblAutoEmail" android:layout_width="wrap_content"
                    android:textSize="9dip" android:layout_height="fill_parent" android:textStyle="italic"
                    android:paddingLeft="8dip" android:text="@string/summary_autoemail"/>
                <TextView android:id="@+id/txtAutoEmail" android:layout_width="wrap_content"
                    android:paddingLeft="3dip" android:textSize="9dip" android:textStyle="italic"
                   android:layout_height="fill_parent"/>
            </TableRow>
        </TableLayout>
        <!--    <TextView android:id="@+id/lblSummary" android:layout_width="fill_parent"
      android:layout_height="wrap_content" android:textStyle="italic"
      android:textSize="11dip" /> -->
    </LinearLayout>
</ScrollView>
```

10.2.2　实现主 Activity

本系统的主 Activity 是 GpsMainActivity，实现文件是 GpsMainActivity.java，具体实现流程如下所示。

（1）定义更新 UI 线程的类 GpsMainActivity，获取 ToggleButton 按钮的开关来显示位置信息，主要实现代码如下所示。

```
public class GpsMainActivity extends Activity implements OnCheckedChangeListener,
         IGpsLoggerServiceClient
{
     /**
      * 用处理器更新 UI 线程
      */
```

```
    public final Handler handler = new Handler();
    private static Intent serviceIntent;
    private GpsLoggingService loggingService;
    /**
     * 提供一个连接到 GPS 记录的服务
     */
    private ServiceConnection gpsServiceConnection = new ServiceConnection()
    {
        public void onServiceDisconnected(ComponentName name)
        {
            loggingService = null;
        }
        public void onServiceConnected(ComponentName name, IBinder service)
        {
            loggingService = ((GpsLoggingService.GpsLoggingBinder) service).getService();
            GpsLoggingService.SetServiceClient(GpsMainActivity.this);
            //设置切换按钮，显示现有的位置信息
            ToggleButton buttonOnOff = (ToggleButton) findViewById(R.id.buttonOnOff);
            if (Session.isStarted())
            {
                buttonOnOff.setChecked(true);
                DisplayLocationInfo(Session.getCurrentLocationInfo());
            }
            buttonOnOff.setOnCheckedChangeListener(GpsMainActivity.this);
        }
    };
```

（2）定义第一次创建样式时触发的方法，具体实现代码如下所示。

```
    /**
     * 第一次创建样式时触发的事件
     */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        String lang = prefs.getString("locale_override", "");
        if (!lang.equalsIgnoreCase(""))
        {
            Locale locale = new Locale(lang);
            Locale.setDefault(locale);
            Configuration config = new Configuration();
            config.locale = locale;
            getBaseContext().getResources().updateConfiguration(config,
                    getBaseContext().getResources().getDisplayMetrics());
        }
        super.onCreate(savedInstanceState);
        Utilities.LogInfo("GPSLogger started");
        setContentView(R.layout.main);
        GetPreferences();
        StartAndBindService();
    }
```

（3）启动定位服务并绑定到当前的 Activity 界面，具体实现代码如下所示。

```
private void StartAndBindService()
{
        Utilities.LogDebug("StartAndBindService - binding now");
        serviceIntent = new Intent(this, GpsLoggingService.class);
        // Start the service in case it isn't already running
        startService(serviceIntent);
        // Now bind to service
        bindService(serviceIntent, gpsServiceConnection, Context.BIND_AUTO_CREATE);
        Session.setBoundToService(true);
}
```

（4）当按钮关闭则停止系统的监听服务，具体实现代码如下所示。

```
private void StopAndUnbindServiceIfRequired()
{
        if(Session.isBoundToService())
        {
                unbindService(gpsServiceConnection);
                Session.setBoundToService(false);
        }
        if(!Session.isStarted())
        {
                Utilities.LogDebug("StopServiceIfRequired - Stopping the service");
                //serviceIntent = new Intent(this, GpsLoggingService.class);
                stopService(serviceIntent);
        }
}

@Override
protected void onPause()
{
        StopAndUnbindServiceIfRequired();
        super.onPause();
}

@Override
protected void onDestroy()
{
        StopAndUnbindServiceIfRequired();
        super.onDestroy();

}
```

（5）当切换按钮被单击时调用方法 onCheckedChanged()，具体实现代码如下所示。

```
public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
{
        if (isChecked)
        {
            GetPreferences();

                loggingService.StartLogging();
        }
```

```
            else
            {
                loggingService.StopLogging();
            }
        }
```

（6）根据用户设置选项值显示一个具有良好可读性的视图界面，具体实现代码如下所示。

```
        private void ShowPreferencesSummary()
        {
            TextView txtLoggingTo = (TextView) findViewById(R.id.txtLoggingTo);
            TextView txtFrequency = (TextView) findViewById(R.id.txtFrequency);
            TextView txtDistance = (TextView) findViewById(R.id.txtDistance);
            TextView txtAutoEmail = (TextView) findViewById(R.id.txtAutoEmail);
            if (!AppSettings.shouldLogToKml() && !AppSettings.shouldLogToGpx())
            {
                txtLoggingTo.setText(R.string.summary_loggingto_screen);
            }
            else if (AppSettings.shouldLogToGpx() && AppSettings.shouldLogToKml())
            {
                txtLoggingTo.setText(R.string.summary_loggingto_both);
            }
            else
            {
                txtLoggingTo.setText((AppSettings.shouldLogToGpx() ? "GPX" : "KML"));
            }
            if (AppSettings.getMinimumSeconds() > 0)
            {
                String descriptiveTime = Utilities.GetDescriptiveTimeString(AppSettings.getMinimumSeconds(),
                        getBaseContext());
                txtFrequency.setText(descriptiveTime);
            }
            else
            {
                txtFrequency.setText(R.string.summary_freq_max);
            }
            if (AppSettings.getMinimumDistance() > 0)
            {
                if (AppSettings.shouldUseImperial())
                {
                    int minimumDistanceInFeet = Utilities.MetersToFeet(AppSettings.getMinimumDistance());
                    txtDistance.setText(((minimumDistanceInFeet == 1)
                        ? getString(R.string.foot)
                        : String.valueOf(minimumDistanceInFeet) + getString(R.string.feet)));
                }
                else
                {
                    txtDistance.setText(((AppSettings.getMinimumDistance() == 1)
                        ? getString(R.string.meter)
                        : String.valueOf(AppSettings.getMinimumDistance()) + getString(R.string.meters)));
                }
            }
            else
```

```
            {
                txtDistance.setText(R.string.summary_dist_regardless);
            }
            if (AppSettings.isAutoEmailEnabled())
            {
                String autoEmailResx;
                if (AppSettings.getAutoEmailDelay() == 0)
                {
                    autoEmailResx = "autoemail_frequency_whenistop";
                }
                else
                {

                    autoEmailResx = "autoemail_frequency_"
                            + String.valueOf(AppSettings.getAutoEmailDelay()).replace(".", "");
                    //.replace(".0", "")
                }
                String autoEmailDesc = getString(getResources().getIdentifier(autoEmailResx, "string",
                getPackageName()));

//              String autoEmailDesc = getString(getResources().getIdentifier(
//                      getPackageName() +    ":string/" + autoEmailResx, null, null));

                txtAutoEmail.setText(autoEmailDesc);
            }
            else
            {
                TableRow trAutoEmail = (TableRow) findViewById(R.id.trAutoEmail);
                trAutoEmail.setVisibility(View.INVISIBLE);
            }
        }
```

（7）根据用户选择的菜单项调用不同的处理方法，具体实现代码如下所示。

```
        public boolean onOptionsItemSelected(MenuItem item)
        {
            int itemId = item.getItemId();
            Utilities.LogInfo("Option item selected - " + String.valueOf(item.getTitle()));
            switch (itemId)
            {
                case R.id.mnuSettings:
                    Intent settingsActivity = new Intent(getBaseContext(), GpsSettingsActivity.class);
                    startActivity(settingsActivity);
                    break;
                case R.id.mnuOSM:
                    UploadToOpenStreetMap();
                    break;
                case R.id.mnuAnnotate:
                    Annotate();
                    break;
                case R.id.mnuShare:
                    Share();
                    break;
```

```
                case R.id.mnuEmailnow:
                    EmailNow();
                    break;
                case R.id.mnuExit:
                    loggingService.StopLogging();
                    loggingService.stopSelf();
                    System.exit(0);
                    break;
            }
            return false;
        }
        private void EmailNow()
        {
            if(Utilities.IsEmailSetup(getBaseContext()))
            {
                loggingService.ForceEmailLogFile();
            }
            else
            {
                Intent emailSetup = new Intent(getBaseContext(), AutoEmailActivity.class);
                startActivity(emailSetup);
            }
        }
```

（8）通过方法 Share()实现轨迹分享功能，允许用户发送带位置的 GPX/KML 文件的位置，或者只使用一个提供者，可以使用的分享方式有 Facebook、短信、电子邮件、推特和蓝牙。方法 Share()的具体实现代码如下所示。

```
        private void Share()
        {
            try
            {
                final String locationOnly = getString(R.string.sharing_location_only);
                final File gpxFolder = new File(Environment.getExternalStorageDirectory(), "GPSLogger");
                if (gpxFolder.exists())
                {
                    String[ ] enumeratedFiles = gpxFolder.list();
                    List<String> fileList = new ArrayList<String>(Arrays.asList(enumeratedFiles));
                    Collections.reverse(fileList);
                    fileList.add(0, locationOnly);
                    final String[ ] files = fileList.toArray(new String[0]);
                    final Dialog dialog = new Dialog(this);
                    dialog.setTitle(R.string.sharing_pick_file);
                    dialog.setContentView(R.layout.filelist);
                    ListView thelist = (ListView) dialog.findViewById(R.id.listViewFiles);
                    thelist.setAdapter(new ArrayAdapter<String>(getBaseContext(),
                            android.R.layout.simple_list_item_single_choice, files));
                    thelist.setOnItemClickListener(new OnItemClickListener()
                    {
                        public void onItemClick(AdapterView<?> av, View v, int index, long arg)
                        {
                            dialog.dismiss();
```

```
                        String chosenFileName = files[index];
                        final Intent intent = new Intent(Intent.ACTION_SEND);
                        // intent.setType("text/plain");
                        intent.setType("*/*");
                        if (chosenFileName.equalsIgnoreCase(locationOnly))
                        {
                            intent.setType("text/plain");
                        }
                        intent.putExtra(Intent.EXTRA_SUBJECT, getString(R.string.sharing_mylocation));
                        if (Session.hasValidLocation())
                        {
                            String bodyText = getString(R.string.sharing_latlong_text,
                                    String.valueOf(Session.getCurrentLatitude()),
                                    String.valueOf(Session.getCurrentLongitude()));
                            intent.putExtra(Intent.EXTRA_TEXT, bodyText);
                            intent.putExtra("sms_body", bodyText);
                        }
                        if (chosenFileName.length() > 0
                                && !chosenFileName.equalsIgnoreCase(locationOnly))
                        {
                            intent.putExtra(Intent.EXTRA_STREAM,
                                    Uri.fromFile(new File(gpxFolder, chosenFileName)));
                        }
                        startActivity(Intent.createChooser(intent, getString(R.string.sharing_via)));
                    }
                });
                dialog.show();
            }
            else
            {
                Utilities.MsgBox(getString(R.string.sorry), getString(R.string.no_files_found), this);
            }
        }
        catch (Exception ex)
        {
            Utilities.LogError("Share", ex);
        }
    }
```

（9）编写方法 UploadToOpenStreetMap()上传一个跟踪 GPS 记录的对象，具体实现代码如下所示。

```
    private void UploadToOpenStreetMap()
    {
        if(!Utilities.IsOsmAuthorized(getBaseContext()))
        {
            startActivity(Utilities.GetOsmSettingsIntent(getBaseContext()));
            return;
        }
        final String goToOsmSettings = getString(R.string.menu_settings);

        final File gpxFolder = new File(Environment.getExternalStorageDirectory(), "GPSLogger");
        if (gpxFolder.exists())
        {
```

```
            FilenameFilter select = new FilenameFilter()
            {

                public boolean accept(File dir, String filename)
                {
                  return filename.toLowerCase().contains(".gpx");
                }
            };
            String[ ] enumeratedFiles = gpxFolder.list(select);
            List<String> fileList = new ArrayList<String>(Arrays.asList(enumeratedFiles));
            Collections.reverse(fileList);
            fileList.add(0, goToOsmSettings);
            final String[ ] files = fileList.toArray(new String[0]);
            final Dialog dialog = new Dialog(this);
            dialog.setTitle(R.string.osm_pick_file);
            dialog.setContentView(R.layout.filelist);
            ListView thelist = (ListView) dialog.findViewById(R.id.listViewFiles);
            thelist.setAdapter(new ArrayAdapter<String>(getBaseContext(),
                    android.R.layout.simple_list_item_single_choice, files));
            thelist.setOnItemClickListener(new OnItemClickListener()
            {
                public void onItemClick(AdapterView<?> av, View v, int index, long arg)
                {

                    dialog.dismiss();
                    String chosenFileName = files[index];

                    if(chosenFileName.equalsIgnoreCase(goToOsmSettings))
                    {
                        startActivity(Utilities.GetOsmSettingsIntent(getBaseContext()));
                    }
                    else
                    {
                        OSMHelper osm = new OSMHelper(GpsMainActivity.this);
                        Utilities.ShowProgress(GpsMainActivity.this, getString(R.string.osm_uploading),
                        getString(R.string.please_wait));
                        osm.UploadGpsTrace(chosenFileName);
                    }
                }
            });
            dialog.show();
        }
        else
        {
            Utilities.MsgBox(getString(R.string.sorry), getString(R.string.no_files_found), this);
        }
    }
```

（10）通过方法 Annotate()提示用户输入，然后添加文本日志文件，具体实现代码如下所示。

```
    private void Annotate()
    {
        if (!AppSettings.shouldLogToGpx() && !AppSettings.shouldLogToKml())
```

```
        {
            return;
        }

        if (!Session.shoulAllowDescription())
        {
            Utilities.MsgBox(getString(R.string.not_yet),
                    getString(R.string.cant_add_description_until_next_point),
                    GetActivity());
            return;
        }
        AlertDialog.Builder alert = new AlertDialog.Builder(GpsMainActivity.this);
        alert.setTitle(R.string.add_description);
        alert.setMessage(R.string.letters_numbers);
        //设置一个 EditText 视图用来获取用户的输入
        final EditText input = new EditText(getBaseContext());
        alert.setView(input);
        alert.setPositiveButton(R.string.ok, new DialogInterface.OnClickListener()
        {
            public void onClick(DialogInterface dialog, int whichButton)
            {
                final String desc = Utilities.CleanDescription(input.getText().toString());
                Annotate(desc);
            }
        });
        alert.setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener()
        {
            public void onClick(DialogInterface dialog, int whichButton)
            {
                // Cancelled
            }
        });
        alert.show();
    }
```

（11）编写方法 ClearForm()清理当前屏幕视图，并删除所有获取的值。具体实现代码如下所示。

```
    public void ClearForm()
    {
        TextView tvLatitude = (TextView) findViewById(R.id.txtLatitude);
        TextView tvLongitude = (TextView) findViewById(R.id.txtLongitude);
        TextView tvDateTime = (TextView) findViewById(R.id.txtDateTimeAndProvider);
        TextView tvAltitude = (TextView) findViewById(R.id.txtAltitude);
        TextView txtSpeed = (TextView) findViewById(R.id.txtSpeed);
        TextView txtSatellites = (TextView) findViewById(R.id.txtSatellites);
        TextView txtDirection = (TextView) findViewById(R.id.txtDirection);
        TextView txtAccuracy = (TextView) findViewById(R.id.txtAccuracy);
        tvLatitude.setText("");
        tvLongitude.setText("");
        tvDateTime.setText("");
        tvAltitude.setText("");
        txtSpeed.setText("");
        txtSatellites.setText("");
```

```
            txtDirection.setText("");
            txtAccuracy.setText("");
        }
```

（12）在顶部的状态标签设置信息，具体实现代码如下所示。

```
        private void SetStatus(String message)
        {
            TextView tvStatus = (TextView) findViewById(R.id.textStatus);
            tvStatus.setText(message);
            Utilities.LogInfo(message);
        }
```

（13）设置表中的卫星视图，具体实现代码如下所示。

```
    private void SetSatelliteInfo(int number)
    {
        Session.setSatelliteCount(number);
        TextView txtSatellites = (TextView) findViewById(R.id.txtSatellites);
        txtSatellites.setText(String.valueOf(number));
    }
```

（14）处理指定的位置坐标，并将结果显示在视图中。具体实现代码如下所示。

```
        private void DisplayLocationInfo(Location loc)
        {
            try
            {
                if (loc == null)
                {
                    return;
                }
                Session.setLatestTimeStamp(System.currentTimeMillis());
                TextView tvLatitude = (TextView) findViewById(R.id.txtLatitude);
                TextView tvLongitude = (TextView) findViewById(R.id.txtLongitude);
                TextView tvDateTime = (TextView) findViewById(R.id.txtDateTimeAndProvider);
                TextView tvAltitude = (TextView) findViewById(R.id.txtAltitude);
                TextView txtSpeed = (TextView) findViewById(R.id.txtSpeed);
                TextView txtSatellites = (TextView) findViewById(R.id.txtSatellites);
                TextView txtDirection = (TextView) findViewById(R.id.txtDirection);
                TextView txtAccuracy = (TextView) findViewById(R.id.txtAccuracy);
                String providerName = loc.getProvider();
                if (providerName.equalsIgnoreCase("gps"))
                {
                    providerName = getString(R.string.providername_gps);
                }
                else
                {
                    providerName = getString(R.string.providername_celltower);
                }
                tvDateTime.setText(new Date().toLocaleString()
                        + getString(R.string.providername_using, providerName));
                tvLatitude.setText(String.valueOf(loc.getLatitude()));
                tvLongitude.setText(String.valueOf(loc.getLongitude()));
                if (loc.hasAltitude())
                {
```

```
            double altitude = loc.getAltitude();
            if (AppSettings.shouldUseImperial())
            {
                tvAltitude.setText(String.valueOf(Utilities.MetersToFeet(altitude))
                        + getString(R.string.feet));
            }
            else
            {
                tvAltitude.setText(String.valueOf(altitude) + getString(R.string.meters));
            }
        }
        else
        {
            tvAltitude.setText(R.string.not_applicable);
        }               if (loc.hasSpeed())
        {
            float speed = loc.getSpeed();
            if (AppSettings.shouldUseImperial())
            {
                txtSpeed.setText(String.valueOf(Utilities.MetersToFeet(speed))
                        + getString(R.string.feet_per_second));
            }
            else
            {
                txtSpeed.setText(String.valueOf(speed) + getString(R.string.meters_per_second));
            }
        }
        else
        {
            txtSpeed.setText(R.string.not_applicable);
        }
        if (loc.hasBearing())
        {
            float bearingDegrees = loc.getBearing();
            String direction;
            direction = Utilities.GetBearingDescription(bearingDegrees, getBaseContext());
            txtDirection.setText(direction + "(" + String.valueOf(Math.round(bearingDegrees))
                    + getString(R.string.degree_symbol) + ")");
        }
        else
        {
            txtDirection.setText(R.string.not_applicable);
        }
        if (!Session.isUsingGps())
        {
            txtSatellites.setText(R.string.not_applicable);
            Session.setSatelliteCount(0);
        }
        if (loc.hasAccuracy())
        {
            float accuracy = loc.getAccuracy();
```

```
                    if (AppSettings.shouldUseImperial())
                    {
                        txtAccuracy.setText(getString(R.string.accuracy_within,
                                String.valueOf(Utilities.MetersToFeet(accuracy)), getString(R.string.feet)));
                    }
                    else
                    {
                        txtAccuracy.setText(getString(R.string.accuracy_within, String.valueOf(accuracy),
                                getString(R.string.meters)));
                    }
                }
                else
                {
                    txtAccuracy.setText(R.string.not_applicable);
                }
            }
            catch (Exception ex)
            {
                SetStatus(getString(R.string.error_displaying, ex.getMessage()));
            }
        }
```

在主 Activity 的实现过程中用到了系统服务 Activity，其实现文件是 GpsLoggingService.java，功能是提供了本系统所需要的后台服务方法。文件 GpsLoggingService.java 的具体实现流程如下所示。

（1）定义可以调用的类和方法，具体实现代码如下所示。

```
class GpsLoggingBinder extends Binder
{
    public GpsLoggingService getService()
    {
        Utilities.LogDebug("GpsLoggingBinder.getService");
        return GpsLoggingService.this;
    }
}
```

（2）建立基于用户偏好设置的电邮自动计时器，具体实现代码如下所示。

```
private void SetupAutoEmailTimers()
{
    Utilities.LogDebug("GpsLoggingService.SetupAutoEmailTimers");
    Utilities.LogDebug("isAutoEmailEnabled - " + String.valueOf(AppSettings.isAutoEmailEnabled()));
    Utilities.LogDebug("Session.getAutoEmailDelay - " + String.valueOf(Session.getAutoEmailDelay()));
    if (AppSettings.isAutoEmailEnabled() && Session.getAutoEmailDelay() > 0)
    {
        Utilities.LogDebug("Setting up email alarm");
        long triggerTime = System.currentTimeMillis()
                + (long) (Session.getAutoEmailDelay() * 60 * 60 * 1000);
        alarmIntent = new Intent(getBaseContext(), AlarmReceiver.class);
        PendingIntent sender = PendingIntent.getBroadcast(this, 0, alarmIntent,
                PendingIntent.FLAG_UPDATE_CURRENT);
        AlarmManager am = (AlarmManager) getSystemService(ALARM_SERVICE);
        am.set(AlarmManager.RTC_WAKEUP, triggerTime, sender);
    }
    else
```

```
    {
        Utilities.LogDebug("Checking if alarmIntent is null");
        if (alarmIntent != null)
        {
            Utilities.LogDebug("alarmIntent was null, canceling alarm");
            CancelAlarm();
        }
    }
}
```

（3）如果调用用户选择自动邮件日志文件方法则停止记录操作，具体实现代码如下所示。

```
private void AutoEmailLogFileOnStop()
{
    Utilities.LogDebug("GpsLoggingService.AutoEmailLogFileOnStop");
     Utilities.LogVerbose("isAutoEmailEnabled - " + AppSettings.isAutoEmailEnabled());
    // autoEmailDelay 0 means send it when you stop logging.
    if (AppSettings.isAutoEmailEnabled() && Session.getAutoEmailDelay() == 0)
    {
        Session.setEmailReadyToBeSent(true);
        AutoEmailLogFile();
    }
}
```

（4）调用自动电子邮件辅助处理文件并将其发送，具体实现代码如下所示。

```
private void AutoEmailLogFile()
{
    Utilities.LogDebug("GpsLoggingService.AutoEmailLogFile");
    Utilities.LogVerbose("isEmailReadyToBeSent - " + Session.isEmailReadyToBeSent());
    if (Session.getCurrentFileName() != null && Session.getCurrentFileName().length() > 0
            && Session.isEmailReadyToBeSent())
    {
        if(IsMainFormVisible())
        {
            Utilities.ShowProgress(mainServiceClient.GetActivity(), getString(R.string.autoemail_sending),
                getString(R.string.please_wait));
        }
        Utilities.LogInfo("Emailing Log File");
        AutoEmailHelper aeh = new AutoEmailHelper(GpsLoggingService.this);
        aeh.SendLogFile(Session.getCurrentFileName(), false);
        SetupAutoEmailTimers();

        if(IsMainFormVisible())
        {
            Utilities.HideProgress();
        }
    }
}
protected void ForceEmailLogFile()
{

    Utilities.LogDebug("GpsLoggingService.ForceEmailLogFile");
    if (Session.getCurrentFileName() != null && Session.getCurrentFileName().length() > 0)
```

```
    {
        if(IsMainFormVisible())
        {
            Utilities.ShowProgress(mainServiceClient.GetActivity(), getString(R.string.autoemail_sending),
                getString(R.string.please_wait));
        }
        Utilities.LogInfo("Force emailing Log File");
        AutoEmailHelper aeh = new AutoEmailHelper(GpsLoggingService.this);
        aeh.SendLogFile(Session.getCurrentFileName(), true);

        if(IsMainFormVisible())
        {
            Utilities.HideProgress();
        }
    }
}
```

（5）设置此服务的活动形式，活动形式需要调用 IGpsLoggerServiceClient。具体实现代码如下所示。

```
protected static void SetServiceClient(IGpsLoggerServiceClient mainForm)
{
    mainServiceClient = mainForm;
}
```

（6）根据用户的偏好设置选择并填充 appSettings 对象，并设置电子邮件的定时器。具体实现代码如下所示。

```
private void GetPreferences()
{
    Utilities.LogDebug("GpsLoggingService.GetPreferences");
    Utilities.PopulateAppSettings(getBaseContext());
    Utilities.LogDebug("Session.getAutoEmailDelay: " + Session.getAutoEmailDelay());
    Utilities.LogDebug("AppSettings.getAutoEmailDelay: " + AppSettings.getAutoEmailDelay());
    if (Session.getAutoEmailDelay() != AppSettings.getAutoEmailDelay())
    {
        Utilities.LogDebug("Old autoEmailDelay - " + String.valueOf(Session.getAutoEmailDelay())
                + "; New -" + String.valueOf(AppSettings.getAutoEmailDelay()));
        Session.setAutoEmailDelay(AppSettings.getAutoEmailDelay());
        SetupAutoEmailTimers();
    }
}
```

（7）实现复位处理，具体实现代码如下所示。

```
protected void StartLogging()
{
    Utilities.LogDebug("GpsLoggingService.StartLogging");
    Session.setAddNewTrackSegment(true);
    if (Session.isStarted())
    {
        return;
    }
    Utilities.LogInfo("Starting logging procedures");
    startForeground(NOTIFICATION_ID, null);
    Session.setStarted(true);
    GetPreferences();
```

```
        Notify();
        ResetCurrentFileName();
        ClearForm();
        StartGpsManager();
}
```

（8）停止记录，删除通知，停止 GPS 记录服务，通过定时器停止邮件。具体实现代码如下所示。

```
protected void StopLogging()
{
        Utilities.LogDebug("GpsLoggingService.StopLogging");
        Session.setAddNewTrackSegment(true);
        Utilities.LogInfo("Stopping logging");
        Session.setStarted(false);
        // Email log file before setting location info to null
        AutoEmailLogFileOnStop();
        CancelAlarm();
        Session.setCurrentLocationInfo(null);
        stopForeground(true);
        RemoveNotification();
        StopGpsManager();
        StopMainActivity();
}
```

（9）状态栏中显示通知，具体实现代码如下所示。

```
private void Notify()
{
        Utilities.LogDebug("GpsLoggingService.Notify");
        if (AppSettings.shouldShowInNotificationBar())
        {
                gpsNotifyManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
                ShowNotification();
        }
        else
        {
                RemoveNotification();
        }
}
```

（10）如果图标是可见的，则隐藏状态栏中的通知，具体实现代码如下所示。

```
private void RemoveNotification()
{
        Utilities.LogDebug("GpsLoggingService.RemoveNotification");
        try
        {
                if (Session.isNotificationVisible())
                {
                        gpsNotifyManager.cancelAll();
                }
        }
        catch (Exception ex)
        {
                Utilities.LogError("RemoveNotification", ex);
        }
```

```
	finally
	{
		// notificationVisible = false;
		Session.setNotificationVisible(false);
	}
}
```

（11）在状态条中显示 GPS 记录器的通知图标，具体实现代码如下所示。

```
private void ShowNotification()
{
	Utilities.LogDebug("GpsLoggingService.ShowNotification");
	// What happens when the notification item is clicked
	Intent contentIntent = new Intent(this, GpsMainActivity.class);
	PendingIntent pending = PendingIntent.getActivity(getBaseContext(), 0, contentIntent,
			android.content.Intent.FLAG_ACTIVITY_NEW_TASK);
	Notification nfc = new Notification(R.drawable.gpsloggericon2, null, System.currentTimeMillis());
	nfc.flags |= Notification.FLAG_ONGOING_EVENT;
	NumberFormat nf = new DecimalFormat("###.######");
	String contentText = getString(R.string.gpslogger_still_running);
	if (Session.hasValidLocation())
	// if (currentLatitude != 0 && currentLongitude != 0)
	{
		contentText = nf.format(Session.getCurrentLatitude()) + ","
				+ nf.format(Session.getCurrentLongitude());
	}
	nfc.setLatestEventInfo(getBaseContext(), getString(R.string.gpslogger_still_running),
			contentText, pending);
	gpsNotifyManager.notify(NOTIFICATION_ID, nfc);
	Session.setNotificationVisible(true);
}
```

（12）根据用户的偏好设置选项启动 GPS 功能，具体实现代码如下所示。

```
private void StartGpsManager()
{
	Utilities.LogDebug("GpsLoggingService.StartGpsManager");
	GetPreferences();
	gpsLocationListener = new GeneralLocationListener(this);
	towerLocationListener = new GeneralLocationListener(this);
	gpsLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
	towerLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
	CheckTowerAndGpsStatus();
	if (Session.isGpsEnabled() && !AppSettings.shouldPreferCellTower())
	{
		Utilities.LogInfo("Requesting GPS location updates");
		// gps satellite based
		gpsLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
				AppSettings.getMinimumSeconds() * 1000, AppSettings.getMinimumDistance(),
				gpsLocationListener);
		gpsLocationManager.addGpsStatusListener(gpsLocationListener);
		Session.setUsingGps(true);
	}
	else if (Session.isTowerEnabled())
```

```
    {
        Utilities.LogInfo("Requesting tower location updates");
        Session.setUsingGps(false);
        // isUsingGps = false;
        // Cell tower and wifi based
        towerLocationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,
                AppSettings.getMinimumSeconds() * 1000, AppSettings.getMinimumDistance(),
                towerLocationListener);
    }
    else
    {
        Utilities.LogInfo("No provider available");
        Session.setUsingGps(false);
        SetStatus(R.string.gpsprovider_unavailable);
        SetFatalMessage(R.string.gpsprovider_unavailable);
        StopLogging();
        return;
    }
    SetStatus(R.string.started);
}
```

（13）周期性检查是否已经启动 GPS 和信号塔，具体实现代码如下所示。

```
private void CheckTowerAndGpsStatus()
{
    Session.setTowerEnabled(towerLocationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER));
    Session.setGpsEnabled(gpsLocationManager.isProviderEnabled(LocationManager.GPS_PROVIDER));
}
```

（14）停止位置管理服务，具体实现代码如下所示。

```
private void StopGpsManager()
{
    Utilities.LogDebug("GpsLoggingService.StopGpsManager");
    if (towerLocationListener != null)
    {
        towerLocationManager.removeUpdates(towerLocationListener);
    }
    if (gpsLocationListener != null)
    {
        gpsLocationManager.removeUpdates(gpsLocationListener);
        gpsLocationManager.removeGpsStatusListener(gpsLocationListener);
    }
    SetStatus(getString(R.string.stopped));
}
```

（15）基于用户偏好设置当前文件名，具体实现代码如下所示。

```
private void ResetCurrentFileName()
{
    Utilities.LogDebug("GpsLoggingService.ResetCurrentFileName");
    String newFileName;
    if (AppSettings.shouldCreateNewFileOnceADay())
    {
        // 20100114.gpx
        SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd");
```

```
            newFileName = sdf.format(new Date());
            Session.setCurrentFileName(newFileName);
        }
        else
        {
            // 201001141833210.gpx
            SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");
            newFileName = sdf.format(new Date());
            Session.setCurrentFileName(newFileName);
        }
        if (IsMainFormVisible())
        {
            mainServiceClient.onFileName(newFileName);
        }
}
```

（16）为客户端显示一个状态信息，具体实现代码如下所示。

```
void SetStatus(String status)
{
        if (IsMainFormVisible())
        {
            mainServiceClient.OnStatusMessage(status);
        }
}
```

到此为止，系统的主 Activity 和服务 Activity 的实现过程介绍完毕，执行后的效果如图 10-2 所示。单击设备中的 MENU 按钮后，在屏幕下方弹出选项设置界面，如图 10-3 所示。



图 10-2　系统主界面

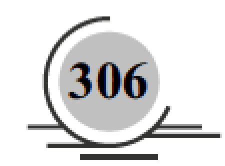


图 10-3　屏幕下方弹出选项设置界面

10.3　系统设置

知识点讲解：光盘:视频\视频讲解\第 10 章\系统设置.avi

当单击图 10-3 中的 Settings 按钮后，会弹出系统设置界面，如图 10-4 所示。



在系统设置界面中，可以设置系统的常用选项参数。本节将详细讲解系统设置模块的实现过程。

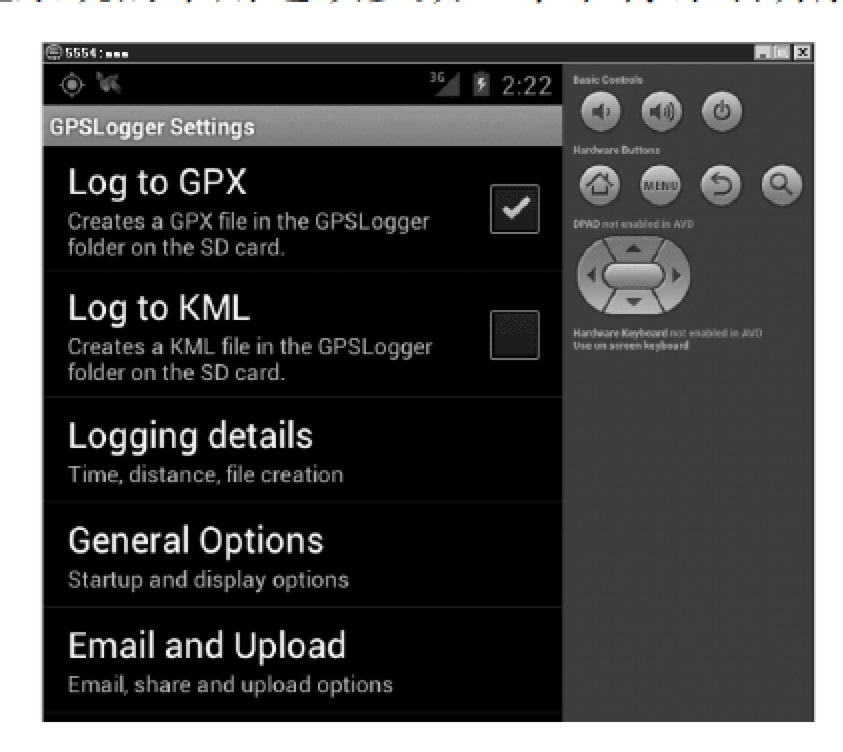


图 10-4　系统设置界面

10.3.1　选项设置

编写文件 AppSettings.java，功能是根据用户各个选项的值来设置系统，例如，设置保存为 GPX（GPS eXchange Format 的缩写，译为 GPS 交换格式，是一个 XML 格式，为应用软体设计的通用 GPS 数据格式，可以用来描述路点、轨迹和路程）、格式数据文件或 KML（是一种文件格式，用于在地球浏览器中显示地理数据，例如 Google 地球、Google 地图和谷歌手机地图）格式数据文件。文件 AppSettings.java 的主要实现代码如下所示。

```
public class AppSettings extends Application
{
	// ---------------------------------------------------
	//用户设置
	// ---------------------------------------------------
	private static boolean useImperial = false;
	private static boolean newFileOnceADay;
	private static boolean preferCellTower;
	private static boolean useSatelliteTime;
	private static boolean logToKml;
	private static boolean logToGpx;
	private static boolean showInNotificationBar;
	private static int minimumDistance;
	private static int minimumSeconds;
	private static String newFileCreation;
	private static Float autoEmailDelay = 0f;
	private static boolean autoEmailEnabled = false;
	private static String smtpServer;
	private static String smtpPort;
	private static String smtpUsername;
	private static String smtpPassword;
	private static String autoEmailTarget;
	private static boolean smtpSsl;
	private static boolean debugToFile;
	public static boolean shouldUseImperial()
	{
		return useImperial;
```

```
}
static void setUseImperial(boolean useImperial)
{
    AppSettings.useImperial = useImperial;
}
/**
 *每天创建一个新文件实现更新
 */
public static boolean shouldCreateNewFileOnceADay()
{
    return newFileOnceADay;
}
static void setNewFileOnceADay(boolean newFileOnceADay)
{
    AppSettings.newFileOnceADay = newFileOnceADay;
}
public static boolean shouldPreferCellTower()
{
    return preferCellTower;
}
static void setPreferCellTower(boolean preferCellTower)
{
    AppSettings.preferCellTower = preferCellTower;
}
public static boolean shouldUseSatelliteTime()
{
    return useSatelliteTime;
}
static void setUseSatelliteTime(boolean useSatelliteTime)
{
    AppSettings.useSatelliteTime = useSatelliteTime;
}
public static boolean shouldLogToKml()
{
    return logToKml;
}
static void setLogToKml(boolean logToKml)
{
    AppSettings.logToKml = logToKml;
}
public static boolean shouldLogToGpx()
{
    return logToGpx;
}
static void setLogToGpx(boolean logToGpx)
{
    AppSettings.logToGpx = logToGpx;
}
public static boolean shouldShowInNotificationBar()
{
    return showInNotificationBar;
```

```
    }
    static void setShowInNotificationBar(boolean showInNotificationBar)
    {
        AppSettings.showInNotificationBar = showInNotificationBar;
    }
    public static int getMinimumDistance()
    {
        return minimumDistance;
    }
    static void setMinimumDistance(int minimumDistance)
    {
        AppSettings.minimumDistance = minimumDistance;
    }
    public static int getMinimumSeconds()
    {
        return minimumSeconds;
    }

    /**
     * @param minimumSeconds
     *            the minimumSeconds to set
     */
    static void setMinimumSeconds(int minimumSeconds)
    {
        AppSettings.minimumSeconds = minimumSeconds;
    }
```

10.3.2　生成 GPX 文件和 KML 文件

在系统设置界面中，可以指定一个文件来保存用户的行走轨迹。本系统提供了两种保存轨迹的文件格式，分别是 GPX 和 KML，如图 10-5 所示。

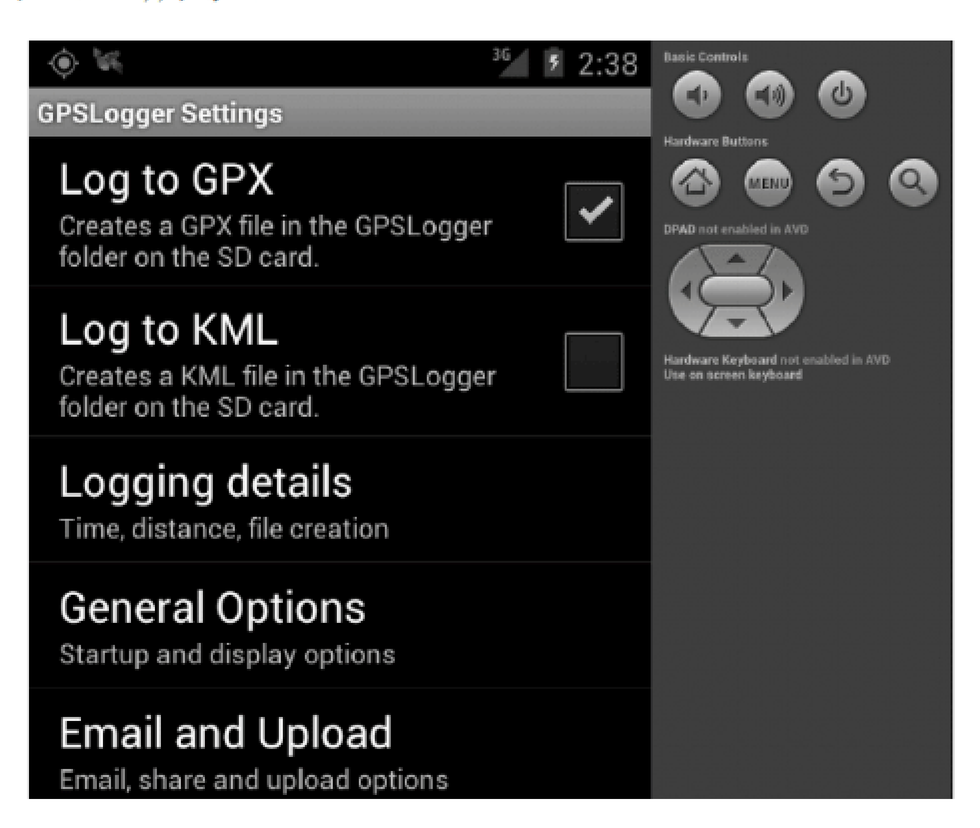


图 10-5　设置保存轨迹的文件格式

编写文件 Gpx10FileLogger.java，生成 GPX 格式的文件，具体实现代码如下所示。

```
class Gpx10FileLogger implements IFileLogger
{
```

```
private final static Object lock = new Object();
private File gpxFile = null;
private boolean useSatelliteTime = false;
private boolean addNewTrackSegment;
private int satelliteCount;

Gpx10FileLogger(File gpxFile, boolean useSatelliteTime, boolean addNewTrackSegment, int satelliteCount)
{
        this.gpxFile = gpxFile;
        this.useSatelliteTime = useSatelliteTime;
        this.addNewTrackSegment = addNewTrackSegment;
        this.satelliteCount = satelliteCount;
}
public void Write(Location loc) throws Exception
{
        try
        {
                Date now;
                if (useSatelliteTime)
                {
                        now = new Date(loc.getTime());
                }
                else
                {
                        now = new Date();
                }
                String dateTimeString = Utilities.GetIsoDateTime(now);

                if (!gpxFile.exists())
                {
                        gpxFile.createNewFile();
                        FileOutputStream initialWriter = new FileOutputStream(gpxFile, true);
                        BufferedOutputStream initialOutput = new BufferedOutputStream(initialWriter);
                        String initialXml = "<?xml version=\"1.0\"?>"
                                + "<gpx version=\"1.0\" creator=\"GPSLogger - http://gpslogger.mendhak.com/\" "
                                + "xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" "
                                + "xmlns=\"http://www.topografix.com/GPX/1/0\" "
                                + "xsi:schemaLocation=\"http://www.topografix.com/GPX/1/0 " "
                                + "http://www.topografix.com/GPX/1/0/gpx.xsd\">"
                                        + "<time>" + dateTimeString + "</time>" + "<bounds />" + "<trk></trk></gpx>" ";
                        initialOutput.write(initialXml.getBytes());
                        initialOutput.flush();
                        initialOutput.close();
                }
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document doc = builder.parse(gpxFile);
            Node trkSegNode;
            NodeList trkSegNodeList = doc.getElementsByTagName("trkseg");
            if(addNewTrackSegment || trkSegNodeList.getLength()==0)
            {
```

```
            NodeList trkNodeList = doc.getElementsByTagName("trk");
            trkSegNode = doc.createElement("trkseg");
            trkNodeList.item(0).appendChild(trkSegNode);
        }
        else
        {
            trkSegNode = trkSegNodeList.item(trkSegNodeList.getLength()-1);
        }
        Element trkptNode = doc.createElement("trkpt");
        Attr latAttribute = doc.createAttribute("lat");
        latAttribute.setValue(String.valueOf(loc.getLatitude()));
        trkptNode.setAttributeNode(latAttribute);
        Attr lonAttribute = doc.createAttribute("lon");
        lonAttribute.setValue(String.valueOf(loc.getLongitude()));
        trkptNode.setAttributeNode(lonAttribute);
        if(loc.hasAltitude())
        {
            Node eleNode = doc.createElement("ele");
            eleNode.appendChild(doc.createTextNode(String.valueOf(loc.getAltitude())));
            trkptNode.appendChild(eleNode);
        }
        Node timeNode = doc.createElement("time");
        timeNode.appendChild(doc.createTextNode(dateTimeString));
        trkptNode.appendChild(timeNode);
        trkSegNode.appendChild(trkptNode);
        if(loc.hasBearing())
        {
            Node courseNode = doc.createElement("course");
            courseNode.appendChild(doc.createTextNode(String.valueOf(loc.getBearing())));
            trkptNode.appendChild(courseNode);
        }
        if(loc.hasSpeed())
        {
            Node speedNode = doc.createElement("speed");
            speedNode.appendChild(doc.createTextNode(String.valueOf(loc.getSpeed())));
            trkptNode.appendChild(speedNode);
        }
        Node srcNode = doc.createElement("src");
        srcNode.appendChild(doc.createTextNode(loc.getProvider()));
        trkptNode.appendChild(srcNode);
        if(Session.getSatelliteCount() > 0)
        {
            Node satNode = doc.createElement("sat");
            satNode.appendChild(doc.createTextNode(String.valueOf(satelliteCount)));
            trkptNode.appendChild(satNode);
        }
        String newFileContents = Utilities.GetStringFromNode(doc);
        synchronized(lock)
        {
            FileOutputStream fos = new FileOutputStream(gpxFile, false);
            fos.write(newFileContents.getBytes());
```

```
                fos.close();
            }
        }
        catch (Exception e)
        {
            Utilities.LogError("Gpx10FileLogger.Write", e);
            throw new Exception("Could not write to GPX file");
        }
    }
    public void Annotate(String description) throws Exception
    {
        if (!gpxFile.exists())
        {
            return;
        }
        try
        {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document doc = builder.parse(gpxFile);
            NodeList trkptNodeList = doc.getElementsByTagName("trkpt");
            Node lastTrkPt = trkptNodeList.item(trkptNodeList.getLength()-1);
            Node nameNode = doc.createElement("name");
            nameNode.appendChild(doc.createTextNode(description));
            lastTrkPt.appendChild(nameNode);
            Node descNode = doc.createElement("desc");
            descNode.appendChild(doc.createTextNode(description));
            lastTrkPt.appendChild(descNode);
            String newFileContents = Utilities.GetStringFromNode(doc);
            synchronized(lock)
            {
                FileOutputStream fos = new FileOutputStream(gpxFile, false);
                fos.write(newFileContents.getBytes());
                fos.close();
            }
        }
        catch(Exception e)
        {
            Utilities.LogError("Gpx10FileLogger.Annotate", e);
            throw new Exception("Could not annotate GPX file");
        }
    }
}
```

编写文件 Kml10FileLogger.java 生成 KML 格式文件，具体实现代码如下所示。

```
class Kml10FileLogger implements IFileLogger
{
    private final static Object lock = new Object();
    private boolean useSatelliteTime;
    private File kmlFile;
    private FileLock kmlLock;
    Kml10FileLogger(File kmlFile, boolean useSatelliteTime)
```

```
{
    this.useSatelliteTime = useSatelliteTime;
    this.kmlFile = kmlFile;
}
public void Write(Location loc) throws Exception
{
    try
    {
        Date now;
        if(useSatelliteTime)
        {
            now = new Date(loc.getTime());
        }
        else
        {
            now = new Date();
        }
        String dateTimeString = Utilities.GetIsoDateTime(now);
        if(!kmlFile.exists())
        {
            kmlFile.createNewFile();
            FileOutputStream initialWriter = new FileOutputStream(kmlFile, true);
            BufferedOutputStream initialOutput = new BufferedOutputStream(initialWriter);
            String initialXml = "<?xml version=\"1.0\"?>"
                    + "<kml xmlns=\"http://www.opengis.net/kml/2.2\"><Document>"
                    + "<Placemark><LineString><extrude>1</extrude><tessellate>1</tessellate>"
                    + "<altitudeMode>absolute</altitudeMode>"
                    + "<coordinates></coordinates></LineString></Placemark>"
                    + "</Document></kml>";
            initialOutput.write(initialXml.getBytes());
            initialOutput.flush();
            initialOutput.close();
        }
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document doc = builder.parse(kmlFile);
        NodeList coordinatesList = doc.getElementsByTagName("coordinates");
        if(coordinatesList.item(0) != null)
        {
            Node coordinates = coordinatesList.item(0);
            Node coordTextNode = coordinates.getFirstChild();
            if(coordTextNode == null)
            {
                coordTextNode = doc.createTextNode("");
                coordinates.appendChild(coordTextNode);
            }
            String coordText = coordinates.getFirstChild().getNodeValue();
            coordText = coordText + "\n" + String.valueOf(loc.getLongitude()) + ","
                    + String.valueOf(loc.getLatitude()) + "," + String.valueOf(loc.getAltitude());
            coordinates.removeChild(coordinates.getFirstChild());
            coordinates.appendChild(doc.createTextNode(coordText));
```

```
            }
            Node documentNode = doc.getElementsByTagName("Document").item(0);
            Node newPlacemark = doc.createElement("Placemark");
            Node timeStamp = doc.createElement("TimeStamp");
            Node whenNode = doc.createElement("when");
            Node whenNodeText = doc.createTextNode(dateTimeString);
            whenNode.appendChild(whenNodeText);
            timeStamp.appendChild(whenNode);
            newPlacemark.appendChild(timeStamp);
            Node newPoint = doc.createElement("Point");
            Node newCoords = doc.createElement("coordinates");
            newCoords.appendChild(doc.createTextNode(String.valueOf(loc.getLongitude()) + ","
                    + String.valueOf(loc.getLatitude()) + "," + String.valueOf(loc.getAltitude())));
            newPoint.appendChild(newCoords);
            newPlacemark.appendChild(newPoint);
            documentNode.appendChild(newPlacemark);
            String newFileContents = Utilities.GetStringFromNode(doc);
            synchronized(lock)
            {
                FileOutputStream fos = new FileOutputStream(kmlFile, false);
                fos.write(newFileContents.getBytes());
                fos.close();
            }
        }
        catch(Exception e)
        {
            Utilities.LogError("Kml10FileLogger.Write", e);
            throw new Exception("Could not write to KML file");
        }
    }
    public void Annotate(String description) throws Exception
    {
        if(!kmlFile.exists())
        {
            return;
        }
        try
        {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document doc = builder.parse(kmlFile);
            NodeList placemarkList = doc.getElementsByTagName("Placemark");
            Node lastPlacemark = placemarkList.item(placemarkList.getLength() - 1);
            Node annotation = doc.createElement("name");
            annotation.appendChild(doc.createTextNode(description));
            lastPlacemark.appendChild(annotation);
            String newFileContents = Utilities.GetStringFromNode(doc);
            synchronized(lock)
            {
                FileOutputStream fos = new FileOutputStream(kmlFile, false);
                fos.write(newFileContents.getBytes());
```

```
                    fos.close();
                }
            }
            catch(Exception e)
            {
                Utilities.LogError("Kml10FileLogger.Annotate", e);
                throw new Exception("Could not annotate KML file");
            }
        }
}
```

10.4　邮件分享提醒

知识点讲解：光盘:视频\视频讲解\第 10 章\邮件分享提醒.avi

在系统设置模块中，可以设置系统邮件来分享行走轨迹信息。邮件分享提醒界面如图 10-6 所示。

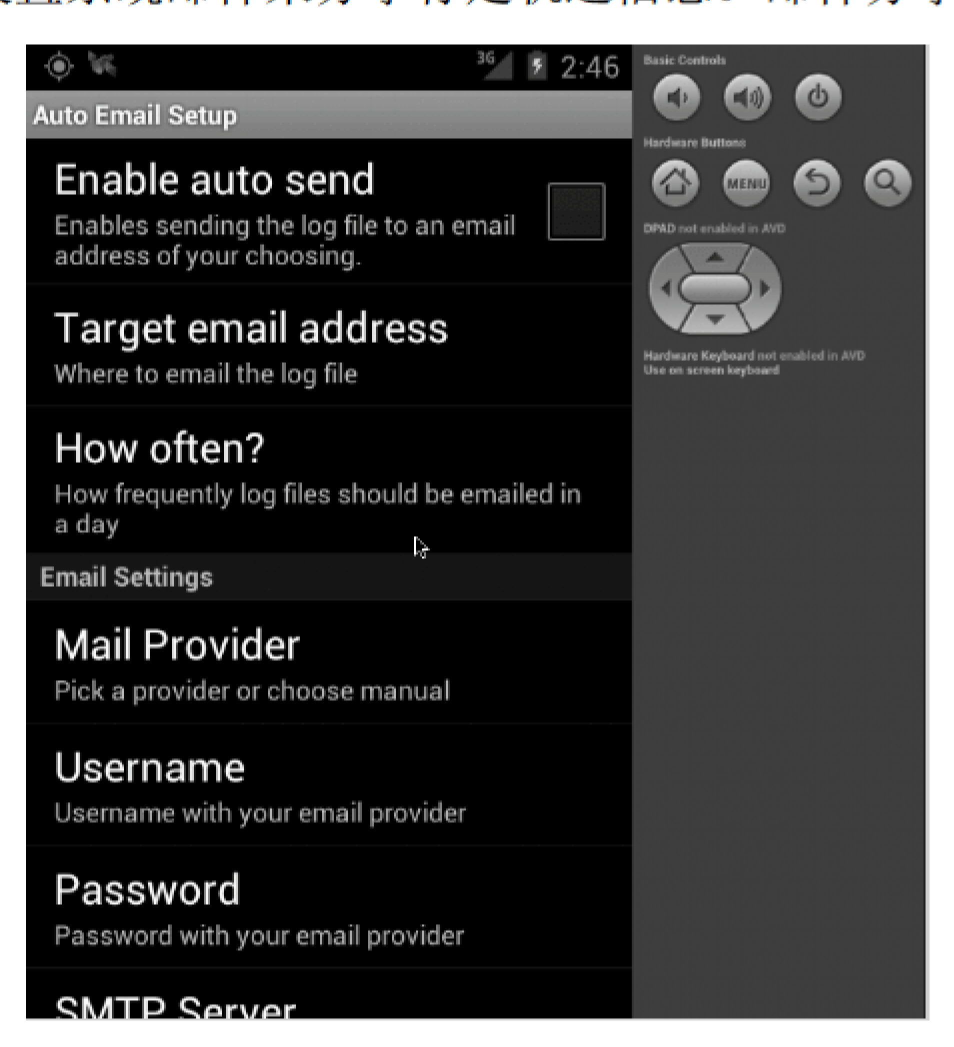


图 10-6　邮件分享提醒界面

10.4.1　基本邮箱设置

编写文件 AutoEmailActivity.java，功能是设置发送邮件的邮箱地址、密码、邮件服务器等信息，这样可以在使用时实现邮件自动发送功能。文件 AutoEmailActivity.java 的具体实现代码如下所示。

```
public class AutoEmailActivity extends PreferenceActivity implements
            OnPreferenceChangeListener, IMessageBoxCallback, IAutoSendHelper,
            OnPreferenceClickListener
{
    private final Handler handler = new Handler();
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
```

```
        addPreferencesFromResource(R.xml.autoemailsettings);
        CheckBoxPreference chkEnabled = (CheckBoxPreference) findPreference("autoemail_enabled");
        chkEnabled.setOnPreferenceChangeListener(this);
        ListPreference lstPresets = (ListPreference) findPreference("autoemail_preset");
        lstPresets.setOnPreferenceChangeListener(this);
        EditTextPreference txtSmtpServer = (EditTextPreference) findPreference("smtp_server");
        EditTextPreference txtSmtpPort = (EditTextPreference) findPreference("smtp_port");
        txtSmtpServer.setOnPreferenceChangeListener(this);
        txtSmtpPort.setOnPreferenceChangeListener(this);
        Preference testEmailPref = (Preference) findPreference("smtp_testemail");
        testEmailPref.setOnPreferenceClickListener(this);
    }
    public boolean onPreferenceClick(Preference preference)
    {
        if (!IsFormValid())
        {
            Utilities.MsgBox(getString(R.string.autoemail_invalid_form),
                    getString(R.string.autoemail_invalid_form_message),
                    AutoEmailActivity.this);
            return false;
        }
        Utilities.ShowProgress(this, getString(R.string.autoemail_sendingtest),
                    getString(R.string.please_wait));
        CheckBoxPreference chkUseSsl = (CheckBoxPreference) findPreference("smtp_ssl");
        EditTextPreference txtSmtpServer = (EditTextPreference) findPreference("smtp_server");
        EditTextPreference txtSmtpPort = (EditTextPreference) findPreference("smtp_port");
        EditTextPreference txtUsername = (EditTextPreference) findPreference("smtp_username");
        EditTextPreference txtPassword = (EditTextPreference) findPreference("smtp_password");
        EditTextPreference txtTarget = (EditTextPreference) findPreference("autoemail_target");
        AutoEmailHelper aeh = new AutoEmailHelper(null);
        aeh.SendTestEmail(txtSmtpServer.getText(), txtSmtpPort.getText(),
                txtUsername.getText(), txtPassword.getText(),
                chkUseSsl.isChecked(), txtTarget.getText(),
                AutoEmailActivity.this, AutoEmailActivity.this);
                return true;
    }
    private boolean IsFormValid()
    {
        CheckBoxPreference chkEnabled = (CheckBoxPreference) findPreference("autoemail_enabled");
        EditTextPreference txtSmtpServer = (EditTextPreference) findPreference("smtp_server");
        EditTextPreference txtSmtpPort = (EditTextPreference) findPreference("smtp_port");
        EditTextPreference txtUsername = (EditTextPreference) findPreference("smtp_username");
        EditTextPreference txtPassword = (EditTextPreference) findPreference("smtp_password");
        EditTextPreference txtTarget = (EditTextPreference) findPreference("autoemail_target");
        if (chkEnabled.isChecked())
        {
            if (txtSmtpServer.getText() != null
                    && txtSmtpServer.getText().length() > 0
                    && txtSmtpPort.getText() != null
                    && txtSmtpPort.getText().length() > 0
                    && txtUsername.getText() != null
```

```
                    && txtUsername.getText().length() > 0
                    && txtPassword.getText() != null
                    && txtPassword.getText().length() > 0
                    && txtTarget.getText() != null
                    && txtTarget.getText().length() > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        return true;
    }
    public boolean onKeyDown(int keyCode, KeyEvent event)
    {
        if (keyCode == KeyEvent.KEYCODE_BACK)
        {
            if (!IsFormValid())
            {
                Utilities.MsgBox(getString(R.string.autoemail_invalid_form),
                        getString(R.string.autoemail_invalid_form_message),
                        this);
                return false;
            }
            else
            {
                return super.onKeyDown(keyCode, event);
            }
        }
        else
        {
            return super.onKeyDown(keyCode, event);
        }
    }
    public void MessageBoxResult(int which)
    {
        finish();
    }
    public boolean onPreferenceChange(Preference preference, Object newValue)
    {
        if (preference.getKey().equals("autoemail_preset"))
        {
            int newPreset = Integer.valueOf(newValue.toString());
            switch (newPreset)
            {
                case 0:
                    // Gmail
                    SetSmtpValues("smtp.gmail.com", "465", true);
                    break;
```

```
                    case 1:
                        // Windows live mail
                        SetSmtpValues("smtp.live.com", "587", false);
                        break;
                    case 2:
                        // Yahoo
                        SetSmtpValues("smtp.mail.yahoo.com", "465", true);
                        break;
                    case 99:
                        // manual
                        break;
                }
            }
            return true;
        }
        private void SetSmtpValues(String server, String port, boolean useSsl)
        {
            SharedPreferences prefs = PreferenceManager
                    .getDefaultSharedPreferences(getBaseContext());
            SharedPreferences.Editor editor = prefs.edit();
            EditTextPreference txtSmtpServer = (EditTextPreference) findPreference("smtp_server");
            EditTextPreference txtSmtpPort = (EditTextPreference) findPreference("smtp_port");
            CheckBoxPreference chkUseSsl = (CheckBoxPreference) findPreference("smtp_ssl");
            // Yahoo
            txtSmtpServer.setText(server);
            editor.putString("smtp_server", server);
            txtSmtpPort.setText(port);
            editor.putString("smtp_port", port);
            chkUseSsl.setChecked(useSsl);
            editor.putBoolean("smtp_ssl", useSsl);
            editor.commit();
        }
        String testResults;
        public void OnRelay(boolean connectionSuccess, String message)
        {
            testResults = message;
            handler.post(showTestResults);
        }
        private final Runnable showTestResults   = new Runnable()
                                                    {
                                                        public void run()
                                                        {
                                                            TestEmailResults();
                                                        }
                                                    };
        private void TestEmailResults()
        {
            Utilities.HideProgress();
            Utilities.MsgBox(getString(R.string.autoemail_testresult_title),
                    testResults, this);
        }
}
```

10.4.2　实现邮件发送功能

编写文件 AutoEmailHelper.java，功能是使用邮件设置模块的邮箱来发送邮件信息，具体实现代码如下所示。

```
public class AutoEmailHelper implements IAutoSendHelper
{
	private GpsLoggingService mainActivity;
	private boolean forcedSend = false;
	public AutoEmailHelper(GpsLoggingService activity)
	{
		this.mainActivity = activity;
	}
	public void SendLogFile(String currentFileName, boolean forcedSend)
	{
		this.forcedSend = forcedSend;
		Thread t = new Thread(new AutoSendHandler(currentFileName, this));
		t.start();
	}
	void SendTestEmail(String smtpServer, String smtpPort,
			String smtpUsername, String smtpPassword, boolean smtpUseSsl,
			String emailTarget, Activity callingActivity, IAutoSendHelper helper)
	{
		Thread t = new Thread(new TestEmailHandler(helper, smtpServer,
				smtpPort, smtpUsername, smtpPassword, smtpUseSsl, emailTarget));
		t.start();
	}
	public void OnRelay(boolean connectionSuccess, String errorMessage)
	{
		if (!connectionSuccess)
		{
			mainActivity.handler.post(mainActivity.updateResultsEmailSendError);
		}
		else
		{
			// 记录邮件发送成功日志信息
			Utilities.LogInfo("Email sent");
			if (!forcedSend)
			{
				Utilities.LogDebug("setEmailReadyToBeSent = false");
				Session.setEmailReadyToBeSent(false);
			}
		}
	}
}
class AutoSendHandler implements Runnable
{
	private String currentFileName;
	private IAutoSendHelper helper;
	public AutoSendHandler(String currentFileName, IAutoSendHelper helper)
```

```
{
        this.currentFileName = currentFileName;
        this.helper = helper;
}
public void run()
{
        File gpxFolder = new File(Environment.getExternalStorageDirectory(),
                "GPSLogger");
        if (!gpxFolder.exists())
        {
                helper.OnRelay(true, null);
                return;
        }
        File gpxFile = new File(gpxFolder.getPath(), currentFileName + ".gpx");
        File kmlFile = new File(gpxFolder.getPath(), currentFileName + ".kml");
        File foundFile = null;
        if (kmlFile.exists())
        {
                foundFile = kmlFile;
        }
        if (gpxFile.exists())
        {
                foundFile = gpxFile;
        }
        if (foundFile == null)
        {
                helper.OnRelay(true, null);
                return;
        }
        String[ ] files = new String[ ]
        { foundFile.getAbsolutePath() };
        File zipFile = new File(gpxFolder.getPath(), currentFileName + ".zip");
        try
        {
                Utilities.LogInfo("Zipping file");
                ZipHelper zh = new ZipHelper(files, zipFile.getAbsolutePath());
                zh.Zip();
                Mail m = new Mail(AppSettings.getSmtpUsername(),
                        AppSettings.getSmtpPassword());
                String[ ] toArr =
                { AppSettings.getAutoEmailTarget() };
                m.setTo(toArr);
                m.setFrom(AppSettings.getSmtpUsername());
                m.setSubject("GPS Log file generated at "
                        + Utilities.GetReadableDateTime(new Date()) + " - "
                        + zipFile.getName());
                m.setBody(zipFile.getName());
                m.setPort(AppSettings.getSmtpPort());
                m.setSecurePort(AppSettings.getSmtpPort());
                m.setSmtpHost(AppSettings.getSmtpServer());
                m.setSsl(AppSettings.isSmtpSsl());
```

```
                m.addAttachment(zipFile.getAbsolutePath());
                Utilities.LogInfo("Sending email...");
                if (m.send())
                {
                    helper.OnRelay(true, "Email was sent successfully.");
                }
                else
                {
                    helper.OnRelay(false, "Email was not sent.");
                }
            }
            catch (Exception e)
            {
                helper.OnRelay(false, e.getMessage());
                Utilities.LogError("AutoSendHandler.run", e);
            }
        }
}

class TestEmailHandler implements Runnable
{
        String smtpServer;
        String smtpPort;
        String smtpUsername;
        String smtpPassword;
        Boolean smtpUseSsl;
        String emailTarget;
        IAutoSendHelper helper;
        public TestEmailHandler(IAutoSendHelper helper, String smtpServer,
                String smtpPort, String smtpUsername, String smtpPassword,
                boolean smtpUseSsl, String emailTarget)
        {
            this.smtpServer = smtpServer;
            this.smtpPort = smtpPort;
            this.smtpPassword = smtpPassword;
            this.smtpUsername = smtpUsername;
            this.smtpUseSsl = smtpUseSsl;
            this.emailTarget = emailTarget;
            this.helper = helper;
        }
        public void run()
        {
            try
            {
                Mail m = new Mail(smtpUsername, smtpPassword);
                String[ ] toArr =
                { emailTarget };
                m.setTo(toArr);
                m.setFrom(smtpUsername);
                m.setSubject("Test Email from GPSLogger at "
                        + Utilities.GetReadableDateTime(new Date()));
```

```
                m.setBody("Test Email from GPSLogger at "
                        + Utilities.GetReadableDateTime(new Date()));
                m.setPort(smtpPort);
                m.setSecurePort(smtpPort);
                m.setSmtpHost(smtpServer);
                m.setSsl(smtpUseSsl);
                m.setDebuggable(true);
                Utilities.LogInfo("Sending email...");
                if (m.send())
                {
                    helper.OnRelay(true, "Email was sent successfully.");
                }
                else
                {
                    helper.OnRelay(false, "Email was not sent.");
                }
            }
            catch (Exception e)
            {
                helper.OnRelay(false, e.getMessage());
                Utilities.LogError("AutoSendHandler.run", e);
            }
        }
}
```

10.5　上传 OSM 地图

知识点讲解：光盘:视频\视频讲解\第 10 章\上传 OSM 地图.avi

OSM 是 OpenStreetMap 的简称，这是一个网上地图协作计划，目的是创造一个内容自由且能让所有用户编辑的世界地图。OSM 的地图由用户根据手提 GPS 装置、航空摄影照片、其他自由内容甚至单靠地方政府、团体或个人绘制。网站里的地图图像及向量数据皆以共享创意姓名标示，用相同方式分享 2.0 授权。OSM 网站的灵感来自维基百科等网站，经注册的用户可上载 GPS 路径及使用内置的编辑程式编辑数据。在上传地图信息之前，需要先获得授权标识。当单击 OpenStreetMap 按钮后会来到授权提示界面，在此界面会显示授权提示信息，如图 10-7 所示。

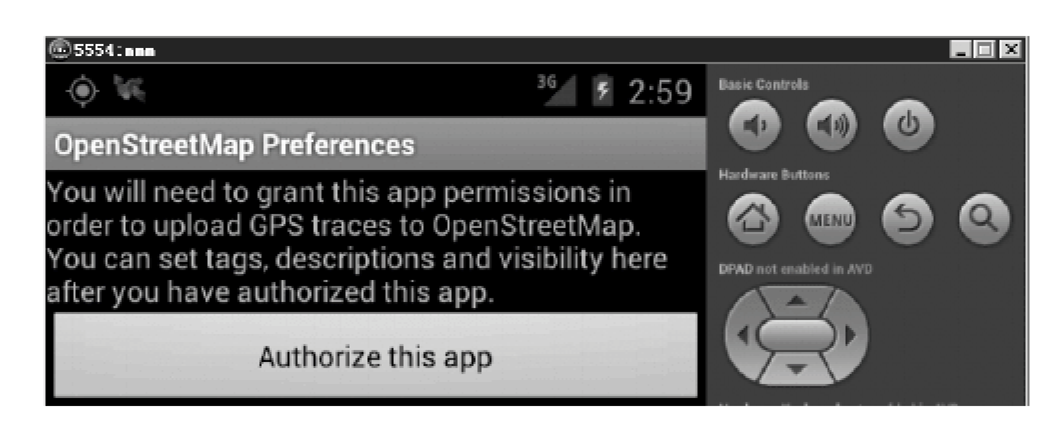


图 10-7　授权提示

10.5.1　授权提示布局文件

授权提示界面的布局文件是 osmauth.xml，具体实现代码如下所示。



```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
      <TableLayout android:id="@+id/TableOSM"
            android:layout_width="fill_parent" android:layout_height="wrap_content"
            android:stretchColumns="1" android:background="#000000">
            <TableRow></TableRow>
            <TableRow>
            <TextView android:id="@+id/lblAuthorizeDescription" android:layout_height="wrap_content"
            android:text="@string/osm_lbl_authorize_description" android:layout_width="wrap_content"></TextView>
         </TableRow>
         <TableRow>
            <Button android:id="@+id/btnAuthorizeOSM"
                    android:text="@string/osm_lbl_authorize" android:layout_height="wrap_content"
                    android:layout_ width ="wrap_content"/>
         </TableRow>
      </TableLayout>
</LinearLayout>
```

通过上述代码在屏幕中显示授权提示信息，并在屏幕下方显示了一个激活按钮。当单击激活按钮后会触发文件 OSMAuthorizationActivity.java，具体实现代码如下所示。

```
public class OSMAuthorizationActivity extends Activity implements
            OnClickListener
{
      private static OAuthProvider provider;
      private static OAuthConsumer consumer;
      @Override
      public void onCreate(Bundle savedInstanceState)
      {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.osmauth);
            final Intent intent = getIntent();
            final Uri myURI = intent.getData();
            if (myURI != null && myURI.getQuery() != null
                        && myURI.getQuery().length() > 0)
            {
                  //User has returned! Read the verifier info from querystring
                  String oAuthVerifier = myURI.getQueryParameter("oauth_verifier");
                  try
                  {
                        SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(getBaseContext());

                        if (provider == null)
                        {
                              provider = Utilities.GetOSMAuthProvider(getBaseContext());
                        }
                        if (consumer == null)
                        {
                              //In case consumer is null, re-initialize from stored values
                              consumer = Utilities.GetOSMAuthConsumer(getBaseContext());
```

```
                }
                //Ask OpenStreetMap for the access token. This is the main event
                provider.retrieveAccessToken(consumer, oAuthVerifier);

                String osmAccessToken = consumer.getToken();
                String osmAccessTokenSecret = consumer.getTokenSecret();

                //Save for use later
                SharedPreferences.Editor editor = prefs.edit();
                editor.putString("osm_accesstoken", osmAccessToken);
                editor.putString("osm_accesstokensecret", osmAccessTokenSecret);
                editor.commit();

                //Now go away
                startActivity(new Intent(getBaseContext(), GpsMainActivity.class));
                finish();
            }
            catch (Exception e)
            {
                Utilities.LogError("OSMAuthorizationActivity.onCreate - user has returned", e);
                Utilities.MsgBox(getString(R.string.sorry), getString(R.string.osm_auth_error), this);
            }
        }
        Button authButton = (Button) findViewById(R.id.btnAuthorizeOSM);
        authButton.setOnClickListener(this);
    }
    public void onClick(View v)
    {
        try
        {
            //User clicks. Set the consumer and provider up
            consumer = Utilities.GetOSMAuthConsumer(getBaseContext());
            provider = Utilities.GetOSMAuthProvider(getBaseContext());
            String authUrl;
            //Get the request token and request token secret
            authUrl = provider.retrieveRequestToken(consumer, OAuth.OUT_OF_BAND);
            //Save for later
            SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(getBaseContext());
            SharedPreferences.Editor editor = prefs.edit();
            editor.putString("osm_requesttoken", consumer.getToken());
            editor.putString("osm_requesttokensecret",consumer.getTokenSecret());
            editor.commit();
            //Open browser, send user to OpenStreetMap.org
            Uri uri = Uri.parse(authUrl);
            Intent intent = new Intent(Intent.ACTION_VIEW, uri);
            startActivity(intent);
        }
        catch (Exception e)
        {
            Utilities.LogError("OSMAuthorizationActivity.onClick", e);
            Utilities.MsgBox(getString(R.string.sorry), getString(R.string.osm_auth_error), this);
```

```
            }
        }
}
```

10.5.2 实现文件上传

编写文件 OSMHelper.java，功能是在获取权限后上传 OpenStreetMap 轨迹文件，具体实现代码如下所示。

```
public class OSMHelper implements IOsmHelper
{
    private GpsMainActivity mainActivity;
    public OSMHelper(GpsMainActivity activity)
    {
        this.mainActivity = activity;
    }

    public void UploadGpsTrace(String fileName)
    {
        File gpxFolder = new File(Environment.getExternalStorageDirectory(), "GPSLogger");
        File chosenFile = new File(gpxFolder, fileName);
        OAuthConsumer consumer = Utilities.GetOSMAuthConsumer(mainActivity.getBaseContext());
        String gpsTraceUrl = mainActivity.getString(R.string.osm_gpstrace_url);

        SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(mainActivity.getBaseContext());
        String description = prefs.getString("osm_description", "");
        String tags = prefs.getString("osm_tags", "");
        String visibility = prefs.getString("osm_visibility", "private");

        Thread t = new Thread(new OsmUploadHandler(this, consumer, gpsTraceUrl, chosenFile,
description, tags, visibility));
        t.start();
    }
    public void OnComplete()
    {
        mainActivity.handler.post(mainActivity.updateOsmUpload);
    }
    private class OsmUploadHandler implements Runnable
    {
        OAuthConsumer consumer;
        String gpsTraceUrl;
        File chosenFile;
        String description;
        String tags;
        String visibility;
        IOsmHelper helper;

        public OsmUploadHandler(IOsmHelper helper, OAuthConsumer consumer, String gpsTraceUrl, File
chosenFile, String description, String tags, String visibility)
        {
            this.consumer = consumer;
            this.gpsTraceUrl = gpsTraceUrl;
```

```
            this.chosenFile = chosenFile;
            this.description = description;
            this.tags = tags;
            this.visibility = visibility;
            this.helper = helper;
        }

        public void run()
        {
            try
            {
               HttpPost request = new HttpPost(gpsTraceUrl);

               consumer.sign(request);
               MultipartEntity entity = new MultipartEntity(HttpMultipartMode.BROWSER_COMPATIBLE);

               FileBody gpxBody = new FileBody(chosenFile);
               entity.addPart("file", gpxBody);
               if(description == null || description.length() <= 0)
               {
                  description = "GPSLogger for Android";
               }

               entity.addPart("description", new StringBody(description));
               entity.addPart("tags", new StringBody(tags));
               entity.addPart("visibility", new StringBody(visibility));

               request.setEntity(entity);
               DefaultHttpClient httpClient = new DefaultHttpClient();
               HttpResponse response = httpClient.execute(request);
               int statusCode = response.getStatusLine().getStatusCode();
               Utilities.LogDebug("OSM Upload - " + String.valueOf(statusCode));
               helper.OnComplete();

            }
            catch(Exception e)
            {
                    Utilities.LogError("OsmUploadHelper.run", e);
            }
        }
    }
}
interface IOsmHelper
{
    public void OnComplete();
}
```

到此为止，本章实例的主要模块介绍完毕。为了节省本书的篇幅，有关本实例其余模块的具体实现过程，请读者参阅本书附带光盘的内容。

第 11 章　智能楼宇灯光控制系统

在本章的实例项目中，将演示在 Android 设备中开发一个智能楼宇灯光控制系统的方法。本实例是一个商业项目，需要开发人员额外编写驱动程序和底层蓝牙控制程序，本章将只讲解 Android 应用程序的实现过程。本实例为智能家居系统开发提供了很好的参考资料和素材，希望大家认真学习，为步入以后工作岗位打下坚实的基础。

11.1　布局文件

知识点讲解：光盘:视频\视频讲解\第 11 章\布局文件.avi

布局文件即 UI 界面设计文件，用于规划在屏幕中显示的控件、图像和文本元素。本节将详细讲解界面布局文件的实现过程。

11.1.1　主布局文件

编写主布局文件 main.xml，设置屏幕中间的动态显示内容，屏幕底部为固定的界面，具体实现代码如下所示。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
     xmlns:custom="http://www.javaeye.com/custom"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/title_relativeLayout"
    android:background="@drawable/one">
     <!-- 中间动态显示界面 -->
     <ViewFlipper android:id="@+id/fliper"
          android:layout_width="fill_parent"
          android:layout_height="fill_parent"
          android:layout_below="@id/title_relativeLayout"
          android:background="#0000"
          android:layout_marginBottom="50.0dip">

     </ViewFlipper>

     <!-- 底部为固定的布局 -->
     <RelativeLayout android:orientation="horizontal"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_alignParentBottom="true"
          style="@android:style/ButtonBar"
```

```
            android:background="@drawable/title_background">
        <ImageButton android:background="#0000" android:layout_width="wrap_content"
android:src="@drawable/add" android:id="@+id/imageButton1" android:layout_height="wrap_content"
android:layout_alignParentBottom="true" android:layout_alignParentLeft="true"></ImageButton>
        <ImageButton android:layout_height="wrap_content" android:background="#0000"
android:id="@+id/imageButton2" android:layout_width="wrap_content" android:src="@drawable/menu"
android:layout_alignTop="@+id/imageButton1" android:layout_alignParentRight="true"></ImageButton>


        </RelativeLayout>
</RelativeLayout>
```

11.1.2 实现蓝牙控制界面

编写文件 bluetooth.xml，通过按钮控件、ImageView 控件和 ToggleButton 控件实现蓝牙控制界面，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="@drawable/one">
       <RelativeLayout  android:layout_width="match_parent"  android:layout_height="match_parent"  android:id=
"@+id/relativeLayout1">
            <ListView android:id="@+id/lvDevices" android:layout_height="wrap_content"
            android:layout_width="match_parent" android:layout_alignParentLeft="true"
            android:layout_alignParentBottom="true" android:layout_below="@+id/btnExit"></ListView>
            <Button android:text="Native Bluetooth visibility" android:layout_height="wrap_content"
            android:layout_width="wrap_content" android:textSize="25sp" android:id="@+id/btnDis"
            android:layout_alignParentTop="true" android:layout_alignRight="@+id/imageView1"
            android:layout_marginRight="64dp" android:layout_marginTop="315dp"></Button>
            <Button android:layout_width="wrap_content" android:textSize="25sp" android:id="@+id/btnExit"
            android:layout_height="wrap_content" android:text="Return to home menu"
            android:layout_below="@+id/btnSearch" android:layout_alignRight="@+id/btnSearch"
            android:layout_marginTop="37dp" android:layout_alignLeft="@+id/btnDis"></Button>
            <Button android:layout_width="110dip" android:textSize="25sp" android:id="@+id/btnSearch"
            android:layout_height="50dip" android:text="Search Equipment"
             android:layout_alignBottom="@+id/btnDis"
            android:layout_alignLeft="@+id/tbtnSwitch"></Button>
            <ToggleButton android:textOff="Off" android:layout_width="110dip" android:layout_height="50dip"
            android:textOn="On" android:id="@+id/tbtnSwitch" android:textSize="23sp" android:text="OFF"
            android:layout_alignBottom="@+id/imageView1" android:layout_alignRight="@+id/imageView2"
            android:layout_marginRight="52dp" android:layout_marginBottom="23dp"></ToggleButton>
            <ImageView android:layout_height="wrap_content" android:layout_width="wrap_content"
            android:id="@+id/imageView1" android:src="@drawable/lanya" android:layout_above="@+id/btnSearch"
            android:layout_toLeftOf="@+id/tbtnSwitch" android:layout_marginRight="41dp"></ImageView>
            <ImageView android:layout_height="wrap_content" android:layout_width="wrap_content"
            android:id="@+id/imageView2" android:src="@drawable/sousuo" android:layout_above="@+id/tbtnSwitch"
            android:layout_centerHorizontal="true"></ImageView>
       </RelativeLayout>
</LinearLayout>
```

11.1.3　显示公司介绍信息

编写文件 company.xml，功能是显示公司介绍信息，具体实现代码如下所示。

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:background="@drawable/bac"
  android:layout_height="match_parent">
    <RelativeLayout android:id="@+id/relativeLayout1" android:layout_width="match_parent"
    android:layout_height="match_parent">
        <ImageView android:src="@drawable/company" android:id="@+id/imageView1"
        android:layout_height="wrap_content" android:layout_width="wrap_content"
        android:layout_alignParentBottom="true" android:layout_centerHorizontal="true"></ImageView>
        <ImageButton android:src="@drawable/back" android:background="#0000"
        android:layout_height="wrap_content" android:id="@+id/back" android:layout_width="wrap_content"
        android:layout_alignParentBottom="true" android:layout_alignParentRight="true"></ImageButton>
        <ImageView android:src="@drawable/ctitle" android:id="@+id/imageView2"
        android:layout_height="wrap_content" android:layout_width="wrap_content"
        android:layout_alignParentTop="true" android:layout_alignRight="@+id/imageView1"
        android:layout_marginRight="22dp"></ImageView>
    </RelativeLayout>
</LinearLayout>
```

11.1.4　系统功能介绍

编写文件 dialog.xml，功能是实现一个系统功能介绍对话框效果，具体实现代码如下所示。

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:id="@+id/dialog" android:weightSum="1">
    <RelativeLayout android:id="@+id/relativeLayout1" android:layout_width="match_parent"
android:layout_height="wrap_content" android:layout_weight="1.09">
        <TextView android:text="@string/intr1" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/textView2"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_below="@+id/textView1"
android:layout_alignLeft="@+id/textView1" android:layout_marginLeft="24dp"></TextView>
        <TextView android:text="@string/intr0" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/textView1"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_alignParentTop="true"
android:layout_alignParentLeft="true" android:layout_marginLeft="50dp"
android:layout_marginTop="53dp"></TextView>
        <TextView android:text="@string/intr2" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/textView3"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_below="@+id/textView2"
android:layout_alignLeft="@+id/textView1"></TextView>
        <TextView android:text="@string/intr3" android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content" android:id="@+id/textView4"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_below="@+id/textView3"
android:layout_alignLeft="@+id/textView2"></TextView>
        <ImageView android:id="@+id/imageView1" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:src="@drawable/main_add_enable"
android:layout_below="@+id/textView4" android:layout_alignLeft="@+id/textView3"
android:layout_marginTop="37dp"></ImageView>
        <TextView android:text="@string/intr4" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/textView5"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_alignBottom="@+id/imageView1"
android:layout_toRightOf="@+id/imageView1" android:layout_marginLeft="34dp"></TextView>
        <ImageView android:id="@+id/imageView2" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:src="@drawable/main_menu_enable"
android:layout_below="@+id/imageView1" android:layout_alignLeft="@+id/imageView1"
android:layout_marginTop="24dp"></ImageView>
        <TextView android:text="@string/intr5" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/textView6"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_alignBottom="@+id/imageView2"
android:layout_alignLeft="@+id/textView5"></TextView>
        <ImageView android:id="@+id/imageView3" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:src="@drawable/main_back_enable"
android:layout_below="@+id/imageView2" android:layout_alignLeft="@+id/imageView2"
android:layout_marginTop="30dp"></ImageView>
        <TextView android:text="@string/intr6" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/textView7"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_alignBottom="@+id/imageView3"
android:layout_alignLeft="@+id/textView6"></TextView>
        <TextView android:text="@string/intr7" android:textColor="#00ff00"
android:layout_width="wrap_content" android:layout_height="wrap_content" android:id="@+id/textView8"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_below="@+id/imageView3"
android:layout_alignLeft="@+id/imageView3" android:layout_marginTop="52dp"></TextView>
        <TextView android:text="@string/intr8" android:textColor="#00ff00"
android:layout_width="wrap_content" android:layout_height="wrap_content" android:id="@+id/textView9"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_below="@+id/textView8"
android:layout_alignLeft="@+id/textView8" android:layout_marginTop="47dp"></TextView>
        <TextView android:text="@string/intr9" android:textColor="#00ff00"
android:layout_width="wrap_content" android:layout_height="wrap_content" android:id="@+id/textView10"
android:textAppearance="?android:attr/textAppearanceLarge" android:layout_below="@+id/textView9"
android:layout_alignLeft="@+id/textView9" android:layout_marginTop="45dp"></TextView>
    </RelativeLayout>
</LinearLayout>
```

11.1.5 第一路调光设置界面

编写文件 first.xml，功能是通过单选按钮列表实现第一路调光设置界面，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#0000"
    android:weightSum="1" android:orientation="vertical">
```



```
    <ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/help_2"
    android:background="#0000"
    android:id="@+id/imageButton1"
    android:layout_gravity="right"></ImageButton>


    <TextView
    android:id="@+id/textview1"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentRight="true"
    android:layout_marginRight="76dp"
    android:layout_marginTop="15dp"></TextView>
    <RelativeLayout
    android:id="@+id/relativeLayout1"
    android:layout_width="match_parent" android:layout_height="696dp" android:gravity="left">
        <RadioGroup android:layout_height="wrap_content" android:id="@+id/radioGroup1"
android:layout_width="wrap_content" android:layout_alignParentTop="true"
android:layout_toRightOf="@+id/imageButton2" android:layout_marginLeft="18dp"
android:layout_marginTop="32dp">
        </RadioGroup>
        <ImageButton android:id="@+id/imageButton5" android:layout_height="wrap_content"
android:background="#0000" android:layout_width="wrap_content" android:src="@drawable/button2_c"
android:layout_alignTop="@+id/imageButton3"
android:layout_alignLeft="@+id/imageButton4"></ImageButton>
        <RadioGroup android:layout_height="wrap_content" android:id="@+id/radioGroup3"
android:layout_width="wrap_content" android:layout_alignTop="@+id/radioGroup2"
android:layout_toRightOf="@+id/radioGroup1" android:layout_marginLeft="45dp">
            <RadioButton android:text="0%" android:layout_height="wrap_content"
android:id="@+id/radio21" android:layout_width="wrap_content" ></RadioButton>
            <RadioButton android:text="20%" android:layout_height="wrap_content"
android:id="@+id/radio22" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="40%" android:layout_height="wrap_content"
android:id="@+id/radio23" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="60%" android:layout_height="wrap_content"
android:id="@+id/radio24" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="80%" android:layout_height="wrap_content"
android:id="@+id/radio25" android:layout_width="wrap_content"></RadioButton>
            <RadioButton android:text="100%" android:layout_height="wrap_content"
android:id="@+id/radio26" android:layout_width="wrap_content"></RadioButton>
        </RadioGroup>
        <ImageButton android:id="@+id/imageButton4" android:layout_height="wrap_content"
android:background="#0000" android:layout_width="wrap_content" android:src="@drawable/button2_o"
android:layout_alignTop="@+id/imageButton2" android:layout_toRightOf="@+id/radioGroup1"
android:layout_marginLeft="24dp"></ImageButton>
        <RadioGroup android:layout_height="wrap_content" android:id="@+id/radioGroup4"
android:layout_width="wrap_content" android:layout_alignTop="@+id/radioGroup3"
```

```
android:layout_toRightOf="@+id/imageButton4" android:layout_marginLeft="76dp">
                    <RadioButton android:text="0%" android:layout_height="wrap_content"
android:id="@+id/radio31" android:layout_width="wrap_content"></RadioButton>
                    <RadioButton android:text="20%" android:layout_height="wrap_content"
android:id="@+id/radio32" android:layout_width="wrap_content"></RadioButton>
                    <RadioButton android:text="40%" android:layout_height="wrap_content"
android:id="@+id/radio33" android:layout_width="wrap_content"></RadioButton>
                <RadioButton android:text="60%" android:layout_width="wrap_content" android:id="@+id/radio34"
android:layout_height="wrap_content" android:layout_below="@+id/radioGroup4"
android:layout_alignLeft="@+id/radioGroup4"></RadioButton>
                <RadioButton android:text="80%" android:layout_width="wrap_content" android:id="@+id/radio35"
android:layout_height="wrap_content" android:layout_below="@+id/radioButton4"
android:layout_alignLeft="@+id/radioButton4"></RadioButton>
                <RadioButton android:text="100%" android:layout_width="wrap_content" android:id="@+id/radio36"
android:layout_height="wrap_content" android:layout_below="@+id/radioButton5"
android:layout_alignLeft="@+id/radioButton5"></RadioButton>
</RadioGroup>
                    <TextView android:layout_height="wrap_content" android:text="第三路调光"
android:textColor="#ffffff" android:id="@+id/textView4" android:layout_width="wrap_content"
android:layout_alignTop="@+id/textView2" android:layout_alignLeft="@+id/radioGroup4"></TextView>
                <ImageButton android:layout_width="wrap_content" android:layout_height="wrap_content"
android:src="@drawable/button3_c" android:background="#0000" android:id="@+id/imageButton7"
android:layout_alignTop="@+id/imageButton5"
android:layout_alignLeft="@+id/imageButton6"></ImageButton>
                <ImageButton android:layout_width="wrap_content" android:layout_height="wrap_content"
android:src="@drawable/button3_o" android:background="#0000" android:id="@+id/imageButton6"
android:layout_alignTop="@+id/imageButton4" android:layout_toRightOf="@+id/imageButton4"
android:layout_marginLeft="56dp"></ImageButton>
                <TextView android:layout_width="wrap_content" android:text="第四路调光"
android:layout_height="wrap_content" android:id="@+id/textView5" android:textColor="#ffffff"
android:layout_alignTop="@+id/textView4" android:layout_alignRight="@+id/radioGroup5"></TextView>
                <ImageButton android:src="@drawable/button4_c" android:layout_height="wrap_content"
android:id="@+id/imageButton9" android:background="#0000" android:layout_width="wrap_content"
android:layout_alignTop="@+id/imageButton7"
android:layout_alignLeft="@+id/imageButton8"></ImageButton>
                <ImageButton android:src="@drawable/split" android:layout_height="wrap_content"
android:id="@+id/imageButton10" android:background="#0000" android:layout_width="1000dip"
android:layout_alignTop="@+id/radioGroup2" android:layout_alignRight="@+id/radioGroup4"
android:layout_marginTop="47dp"></ImageButton>
                <ImageButton android:src="@drawable/split" android:layout_height="wrap_content"
android:id="@+id/imageButton11" android:background="#0000" android:layout_width="wrap_content"
android:layout_alignTop="@+id/imageButton10"
android:layout_alignRight="@+id/radioGroup5"></ImageButton>
                <ImageButton android:background="#0000" android:layout_height="wrap_content"
android:src="@drawable/split" android:id="@+id/imageButton12" android:layout_width="wrap_content"
android:layout_below="@+id/imageButton10" android:layout_alignLeft="@+id/radioGroup2"
android:layout_marginTop="35dp"></ImageButton>
                <ImageButton android:background="#0000" android:layout_height="wrap_content"
android:src="@drawable/split" android:id="@+id/imageButton15" android:layout_width="wrap_content"
android:layout_alignTop="@+id/imageButton12"
android:layout_alignRight="@+id/radioGroup5"></ImageButton>
```

```
		<ImageButton android:background="#0000" android:layout_height="wrap_content"
android:src="@drawable/split" android:id="@+id/imageButton13" android:layout_width="wrap_content"
android:layout_below="@+id/imageButton12" android:layout_alignRight="@+id/imageButton4"
android:layout_marginTop="43dp"></ImageButton>
		<ImageButton android:background="#0000" android:layout_height="wrap_content"
android:src="@drawable/split" android:id="@+id/imageButton17" android:layout_width="wrap_content"
android:layout_alignTop="@+id/imageButton13"
android:layout_alignRight="@+id/radioGroup5"></ImageButton>
		<ImageButton android:background="#0000" android:layout_height="wrap_content"
android:src="@drawable/split" android:id="@+id/imageButton18" android:layout_width="wrap_content"
android:layout_below="@+id/imageButton13" android:layout_alignLeft="@+id/imageButton13"
android:layout_marginTop="39dp"></ImageButton>
		<ImageButton android:background="#0000" android:layout_height="wrap_content"
android:src="@drawable/split" android:id="@+id/imageButton19" android:layout_width="wrap_content"
android:layout_alignTop="@+id/imageButton18"
android:layout_alignRight="@+id/radioGroup5"></ImageButton>
		<ImageButton android:id="@+id/imageButton3" android:layout_width="wrap_content"
android:src="@drawable/button1_c" android:layout_height="wrap_content" android:background="#0000"
android:layout_below="@+id/imageButton2" android:layout_alignLeft="@+id/imageButton2"
android:layout_marginTop="30dp"></ImageButton>
		<RadioGroup android:layout_width="wrap_content" android:id="@+id/radioGroup2"
android:layout_height="wrap_content" android:layout_below="@+id/radioGroup1"
android:layout_alignParentLeft="true" android:layout_marginLeft="179dp" android:layout_marginTop="105dp">
			<RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="0%" android:id="@+id/radio11"></RadioButton>
			<RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="20%" android:id="@+id/radio12"></RadioButton>
			<RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="40%" android:id="@+id/radio13"></RadioButton>
			<RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="60%" android:id="@+id/radio14"></RadioButton>
			<RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="80%" android:id="@+id/radio15"></RadioButton>
			<RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="100%" android:id="@+id/radio16"></RadioButton>
		</RadioGroup>
		<TextView android:text="第二路调光" android:textColor="#ffffff" android:layout_width="wrap_content"
android:id="@+id/textView2" android:layout_height="wrap_content" android:layout_alignTop="@+id/textView3"
android:layout_alignLeft="@+id/radioGroup3"></TextView>
		<TextView android:text="第一路调光" android:textColor="#ffffff" android:layout_width="wrap_content"
android:id="@+id/textView3" android:layout_height="wrap_content" android:layout_below="@+id/radioGroup1"
android:layout_alignLeft="@+id/radioGroup2" android:layout_marginTop="44dp"></TextView>
		<TextView  android:text=" 一 "  android:layout_width="wrap_content"  android:id="@+id/textView7"
android:textSize="30sp"  android:layout_height="wrap_content"  android:layout_below="@+id/imageButton11"
android:layout_alignRight="@+id/textView6"></TextView>
		<TextView  android:text=" 调 "  android:layout_width="wrap_content"  android:id="@+id/textView8"
android:textSize="30sp"  android:layout_height="wrap_content"  android:layout_below="@+id/imageButton15"
android:layout_alignRight="@+id/textView7"></TextView>
		<TextView  android:text=" 光 "  android:layout_width="wrap_content"  android:id="@+id/textView9"
android:textSize="30sp"  android:layout_height="wrap_content"  android:layout_below="@+id/imageButton17"
android:layout_alignRight="@+id/textView8"></TextView>
```

```
            <TextView android:text=" 界" android:layout_width="wrap_content"
android:id="@+id/textView10" android:textSize="30sp" android:layout_height="wrap_content"
android:layout_below="@+id/imageButton19" android:layout_alignRight="@+id/textView9"></TextView>
            <ImageButton android:id="@+id/imageButton8" android:layout_width="wrap_content"
android:src="@drawable/button4_o" android:layout_height="wrap_content" android:background="#0000"
android:layout_alignTop="@+id/imageButton6" android:layout_toRightOf="@+id/imageButton6"
android:layout_marginLeft="39dp"></ImageButton>
            <RadioGroup android:layout_width="wrap_content" android:id="@+id/radioGroup5"
android:layout_height="wrap_content" android:layout_alignTop="@+id/radioGroup4"
android:layout_alignRight="@+id/imageButton8">
                <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="0%" android:id="@+id/radio41"></RadioButton>
                <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="20%" android:id="@+id/radio42"></RadioButton>
                <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="40%" android:id="@+id/radio43"></RadioButton>
                <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="60%" android:id="@+id/radio44"></RadioButton>
                <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="80%" android:id="@+id/radio45"></RadioButton>
                <RadioButton android:layout_height="wrap_content" android:layout_width="wrap_content"
android:text="100%" android:id="@+id/radio46"></RadioButton>
            </RadioGroup>
            <ImageButton android:id="@+id/imageButton14" android:layout_width="1000dip"
android:src="@drawable/line" android:layout_height="5dip"
android:layout_below="@+id/radioGroup3"></ImageButton>
            <ImageButton android:layout_width="5dip" android:src="@drawable/line"
android:layout_height="1000dip" android:id="@+id/imageButton16" android:layout_alignParentTop="true"
android:layout_toRightOf="@+id/textView5" android:layout_marginLeft="46dp"></ImageButton>
            <TextView android:textSize="30sp" android:layout_width="wrap_content" android:text="第"
android:id="@+id/textView6" android:layout_height="wrap_content"
android:layout_alignTop="@+id/radioGroup5" android:layout_toRightOf="@+id/imageButton16"
android:layout_marginLeft="24dp"></TextView>
            <ImageButton android:layout_width="wrap_content" android:src="@drawable/button1_add_x"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton2"
android:layout_below="@+id/imageButton14" android:layout_alignRight="@+id/radioGroup2"
android:layout_marginTop="56dp"></ImageButton>
            <TextView android:id="@+id/textView11" android:text=" 面" android:textSize="30sp"
android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_above="@+id/imageButton14" android:layout_alignLeft="@+id/textView10"></TextView>
            <ImageButton android:layout_height="wrap_content" android:background="#0000"
android:id="@+id/btnopen" android:layout_width="wrap_content" android:src="@drawable/allop"
android:layout_alignTop="@+id/imageButton8" android:layout_alignLeft="@+id/textView11"></ImageButton>
            <ImageButton android:layout_height="wrap_content" android:id="@+id/btnclose"
android:layout_width="wrap_content" android:src="@drawable/allcl" android:background="#0000"
android:layout_alignTop="@+id/imageButton9" android:layout_alignLeft="@+id/btnopen"></ImageButton>


    </RelativeLayout>

</LinearLayout>
```



11.1.6　执行主界面

编写文件 home.xml，此文件是系统执行后进入的主界面，具体实现代码如下所示。

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:background="@drawable/bac"
  android:layout_height="match_parent" android:weightSum="1">
    <RelativeLayout android:id="@+id/relativeLayout1" android:layout_width="match_parent"
android:layout_height="682dp">
        <ImageView android:layout_alignParentLeft="true" android:id="@+id/imageView2"
android:src="@drawable/enter1" android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_below="@+id/imageView1"></ImageView>
        <ImageView android:id="@+id/imageView4" android:src="@drawable/enter2"
android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_below="@+id/imageView2" android:layout_alignParentLeft="true"
android:layout_marginLeft="48dp" android:layout_marginTop="86dp"></ImageView>
        <ImageView android:id="@+id/imageView5" android:src="@drawable/enter3"
android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_alignParentBottom="true" android:layout_alignLeft="@+id/imageView4"
android:layout_marginBottom="66dp"></ImageView>
        <ImageView android:id="@+id/imageView7" android:src="@drawable/split"
android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_alignBottom="@+id/imageView5" android:layout_alignLeft="@+id/imageView6"></ImageView>
        <ImageButton android:id="@+id/Enterp" android:background="#0000"
android:src="@drawable/enter0" android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_alignBottom="@+id/imageView6" android:layout_alignLeft="@+id/Enterr"></ImageButton>
        <ImageButton android:id="@+id/Enterc" android:background="#0000"
android:src="@drawable/enter0" android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_above="@+id/imageView7" android:layout_alignLeft="@+id/Enterp"></ImageButton>
        <ImageView android:id="@+id/imageView6" android:src="@drawable/split"
android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_below="@+id/imageView4" android:layout_alignLeft="@+id/imageView3"></ImageView>
        <ImageView android:id="@+id/imageView1" android:src="@drawable/logo"
android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_alignParentTop="true" android:layout_centerHorizontal="true"></ImageView>
        <ImageButton android:background="#0000" android:id="@+id/Enterr"
android:src="@drawable/enter0" android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_alignBottom="@+id/imageView3" android:layout_toRightOf="@+id/imageView1"
android:layout_marginLeft="63dp"></ImageButton>
        <ImageView android:id="@+id/imageView3" android:src="@drawable/split"
android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_below="@+id/imageView2" android:layout_alignRight="@+id/imageView1"
android:layout_marginRight="87dp"></ImageView>

</RelativeLayout>
    <RelativeLayout android:layout_weight="0.95" android:layout_height="wrap_content"
android:id="@+id/relativeLayout2" android:layout_width="match_parent"></RelativeLayout>
    <RelativeLayout android:layout_width="fill_parent"
```

```
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            style="@android:style/ButtonBar"
            android:background="@drawable/title_background">
            <ImageButton android:layout_width="wrap_content" android:layout_height="wrap_content"
android:background="#0000" android:id="@+id/back" android:src="@drawable/back"
android:layout_centerHorizontal="true"
android:layout_alignTop="@+id/imageButton3"></ImageButton></RelativeLayout>
</LinearLayout>
```

11.1.7 不同房间的照明亮度参考值

编写文件 lightstandard.xml，功能是显示不同房间的照明亮度参考值，具体实现代码如下所示。

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:background="@drawable/bac"
  android:id="@+id/stand" android:weightSum="1">
    <RelativeLayout android:id="@+id/relativeLayout1" android:layout_width="match_parent"
android:layout_height="wrap_content" android:layout_weight="1.09">
        <ImageView android:layout_height="wrap_content" android:src="@drawable/twitter1"
android:id="@+id/imageView1" android:background="#0000" android:layout_width="wrap_content"
android:layout_alignParentTop="true" android:layout_alignRight="@+id/imageView2"
android:layout_marginRight="119dp"></ImageView>
        <ImageView android:layout_height="wrap_content" android:src="@drawable/stand1"
android:id="@+id/imageView2" android:layout_width="wrap_content"
android:layout_below="@+id/imageView1" android:layout_alignParentRight="true"
android:layout_marginRight="173dp"></ImageView>
    </RelativeLayout>
</LinearLayout>
```

11.1.8 产品的详细介绍

编写文件 product.xml，功能是显示本产品的详细介绍信息，具体实现代码如下所示。

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:background="@drawable/bac"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
    <RelativeLayout android:id="@+id/relativeLayout1" android:layout_height="match_parent"
android:layout_width="match_parent">
        <ImageButton android:src="@drawable/back" android:background="#0000"
android:layout_height="wrap_content" android:id="@+id/back" android:layout_width="wrap_content"
android:layout_alignParentBottom="true" android:layout_alignParentRight="true"></ImageButton>
        <ImageView android:src="@drawable/product" android:id="@+id/imageView1"
android:layout_height="wrap_content" android:layout_width="wrap_content"
```

```
android:layout_above="@+id/back"  android:layout_alignParentLeft="true"  android:layout_marginLeft="117dp"
android:layout_marginBottom="31dp"></ImageView>
        <ImageView android:src="@drawable/title" android:id="@+id/imageView2"
android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_alignParentTop="true" android:layout_alignLeft="@+id/imageView1"
android:layout_marginLeft="59dp" android:layout_marginTop="42dp"></ImageView>
    </RelativeLayout>
</LinearLayout>
```

11.1.9　五路调光设置界面

编写文件 second.xml，功能是通过单选按钮列表实现一个五路调光设置界面效果，具体实现代码如下所示。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#0000"
    android:weightSum="1" android:orientation="vertical">
    <ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/help_2"
    android:background="#0000"
    android:id="@+id/imageButton1"
    android:layout_gravity="right"></ImageButton>


    <TextView
    android:id="@+id/textview1"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentRight="true"
    android:layout_marginRight="76dp"
    android:layout_marginTop="15dp"></TextView>
    <RelativeLayout
    android:id="@+id/relativeLayout1"
    android:layout_width="match_parent" android:layout_height="696dp" android:gravity="left">
        <RadioGroup android:layout_height="wrap_content" android:id="@+id/radioGroup1"
android:layout_width="wrap_content" android:layout_alignParentTop="true"
android:layout_toRightOf="@+id/imageButton2" android:layout_marginLeft="18dp"
android:layout_marginTop="32dp">
        </RadioGroup>
        <ImageButton android:layout_width="wrap_content" android:id="@+id/imageButton3"
android:background="#0000" android:layout_height="wrap_content" android:src="@drawable/button5_c"
android:layout_below="@+id/imageButton2" android:layout_alignLeft="@+id/imageButton2"
android:layout_marginTop="26dp"></ImageButton>
        <ImageButton android:id="@+id/imageButton5" android:layout_height="wrap_content"
android:background="#0000" android:layout_width="wrap_content" android:src="@drawable/button6_c"
android:layout_alignTop="@+id/imageButton3"
android:layout_alignLeft="@+id/imageButton4"></ImageButton>
```

```
            <RadioGroup android:layout_height="wrap_content" android:id="@+id/radioGroup3"
android:layout_width="wrap_content" android:layout_alignTop="@+id/radioGroup2"
android:layout_toRightOf="@+id/radioGroup1" android:layout_marginLeft="45dp">
                <RadioButton android:text="0%" android:layout_height="wrap_content"
android:id="@+id/radio21" android:layout_width="wrap_content" ></RadioButton>
                <RadioButton android:text="20%" android:layout_height="wrap_content"
android:id="@+id/radio22" android:layout_width="wrap_content"></RadioButton>
                <RadioButton android:text="40%" android:layout_height="wrap_content"
android:id="@+id/radio23" android:layout_width="wrap_content"></RadioButton>
                <RadioButton android:text="60%" android:layout_height="wrap_content"
android:id="@+id/radio24" android:layout_width="wrap_content"></RadioButton>
                <RadioButton android:text="80%" android:layout_height="wrap_content"
android:id="@+id/radio25" android:layout_width="wrap_content"></RadioButton>
                <RadioButton android:text="100%" android:layout_height="wrap_content"
android:id="@+id/radio26" android:layout_width="wrap_content"></RadioButton>
            </RadioGroup>
            <ImageButton android:id="@+id/imageButton4" android:layout_height="wrap_content"
android:background="#0000" android:layout_width="wrap_content" android:src="@drawable/button6_o"
android:layout_alignTop="@+id/imageButton2" android:layout_toRightOf="@+id/radioGroup1"
android:layout_marginLeft="24dp"></ImageButton>
            <RadioGroup android:layout_height="wrap_content" android:id="@+id/radioGroup4"
android:layout_width="wrap_content" android:layout_alignTop="@+id/radioGroup3"
android:layout_toRightOf="@+id/imageButton4" android:layout_marginLeft="76dp">
                <RadioButton android:text="0%" android:layout_height="wrap_content"
android:id="@+id/radio31" android:layout_width="wrap_content"></RadioButton>
                <RadioButton android:text="20%" android:layout_height="wrap_content"
android:id="@+id/radio32" android:layout_width="wrap_content"></RadioButton>
                <RadioButton android:text="40%" android:layout_height="wrap_content"
android:id="@+id/radio33" android:layout_width="wrap_content"></RadioButton>
             <RadioButton android:text="60%" android:layout_width="wrap_content" android:id="@+id/radio34"
android:layout_height="wrap_content" android:layout_below="@+id/radioGroup4"
android:layout_alignLeft="@+id/radioGroup4"></RadioButton>
             <RadioButton android:text="80%" android:layout_width="wrap_content" android:id="@+id/radio35"
android:layout_height="wrap_content" android:layout_below="@+id/radioButton4"
android:layout_alignLeft="@+id/radioButton4"></RadioButton>
             <RadioButton android:text="100%" android:layout_width="wrap_content" android:id="@+id/radio36"
android:layout_height="wrap_content" android:layout_below="@+id/radioButton5"
android:layout_alignLeft="@+id/radioButton5"></RadioButton></RadioGroup>
                  <TextView android:layout_height="wrap_content" android:text="第七路调光"
android:textColor="#ffffff" android:id="@+id/textView4" android:layout_width="wrap_content"
android:layout_alignTop="@+id/textView2" android:layout_alignLeft="@+id/radioGroup4"></TextView>
             <ImageButton android:layout_width="wrap_content" android:layout_height="wrap_content"
android:src="@drawable/button7_c" android:background="#0000" android:id="@+id/imageButton7"
android:layout_alignTop="@+id/imageButton5"
android:layout_alignLeft="@+id/imageButton6"></ImageButton>
             <ImageButton android:layout_width="wrap_content" android:layout_height="wrap_content"
android:src="@drawable/button7_o" android:background="#0000" android:id="@+id/imageButton6"
android:layout_alignTop="@+id/imageButton4" android:layout_toRightOf="@+id/imageButton4"
android:layout_marginLeft="56dp"></ImageButton>
             <TextView android:layout_height="wrap_content" android:text="第八路调光"
android:textColor="#ffffff" android:id="@+id/textView5" android:layout_width="wrap_content"
```

```
android:layout_alignTop="@+id/textView4" android:layout_alignLeft="@+id/radioGroup5"></TextView>
        <ImageButton android:layout_width="wrap_content" android:layout_height="wrap_content"
android:src="@drawable/button8_c" android:background="#0000" android:id="@+id/imageButton9"
android:layout_alignTop="@+id/imageButton7"
android:layout_alignLeft="@+id/imageButton8"></ImageButton>
        <RadioGroup android:id="@+id/radioGroup2" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_below="@+id/radioGroup1"
android:layout_alignParentLeft="true" android:layout_marginLeft="207dp" android:layout_marginTop="84dp">
            <RadioButton android:text="0%" android:id="@+id/radio11"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
            <RadioButton android:text="20%" android:id="@+id/radio12"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
            <RadioButton android:text="40%" android:id="@+id/radio13"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
            <RadioButton android:text="60%" android:id="@+id/radio14"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
            <RadioButton android:text="80%" android:id="@+id/radio15"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
            <RadioButton android:text="100%" android:id="@+id/radio16"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
        </RadioGroup>
        <TextView android:textColor="#ffffff" android:layout_width="wrap_content" android:text="第五路调光"
android:id="@+id/textView3" android:layout_height="wrap_content" android:layout_below="@+id/radioGroup1"
android:layout_alignLeft="@+id/radioGroup2" android:layout_marginTop="31dp"></TextView>
        <ImageButton android:layout_width="wrap_content" android:src="@drawable/button5_o"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton2"
android:layout_below="@+id/radioGroup2" android:layout_alignRight="@+id/radioGroup2"
android:layout_marginTop="76dp"></ImageButton>
        <TextView android:textColor="#ffffff" android:layout_width="wrap_content" android:text="第六路调光"
android:id="@+id/textView2" android:layout_height="wrap_content" android:layout_alignTop="@+id/textView3"
android:layout_alignLeft="@+id/radioGroup3"></TextView>
        <RadioGroup android:id="@+id/radioGroup5" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_alignTop="@+id/radioGroup4"
android:layout_toRightOf="@+id/imageButton6" android:layout_marginLeft="59dp">
            <RadioButton android:text="0%" android:id="@+id/radio41"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
            <RadioButton android:text="20%" android:id="@+id/radio42"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
            <RadioButton android:text="40%" android:id="@+id/radio43"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
            <RadioButton android:text="60%" android:id="@+id/radio44"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
            <RadioButton android:text="80%" android:id="@+id/radio45"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
            <RadioButton android:text="100%" android:id="@+id/radio46"
android:layout_width="wrap_content" android:layout_height="wrap_content"></RadioButton>
        </RadioGroup>
        <ImageButton android:layout_width="wrap_content" android:src="@drawable/button8_o"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton8"
android:layout_alignTop="@+id/imageButton6" android:layout_toRightOf="@+id/imageButton6"
android:layout_marginLeft="39dp"></ImageButton>
```

```
        <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton10"
android:layout_alignTop="@+id/radioGroup3" android:layout_alignRight="@+id/radioGroup3"
android:layout_marginTop="42dp"></ImageButton>
        <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton11"
android:layout_below="@+id/imageButton10" android:layout_alignRight="@+id/radioGroup3"
android:layout_marginTop="43dp"></ImageButton>
        <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton12"
android:layout_below="@+id/imageButton11" android:layout_alignRight="@+id/textView2"
android:layout_marginTop="42dp"></ImageButton>
        <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton13"
android:layout_alignBottom="@+id/radioGroup3" android:layout_alignRight="@+id/radioGroup3"
android:layout_marginBottom="46dp"></ImageButton>
        <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton14"
android:layout_alignTop="@+id/imageButton10" android:layout_alignRight="@+id/textView5"></ImageButton>
        <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton15"
android:layout_alignTop="@+id/imageButton11"
android:layout_alignRight="@+id/radioGroup5"></ImageButton>
        <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton16"
android:layout_alignTop="@+id/imageButton12"
android:layout_alignRight="@+id/radioGroup5"></ImageButton>
        <ImageButton android:layout_width="wrap_content" android:src="@drawable/split"
android:layout_height="wrap_content" android:background="#0000" android:id="@+id/imageButton17"
android:layout_alignTop="@+id/imageButton13"
android:layout_alignRight="@+id/radioGroup5"></ImageButton>
        <ImageButton android:layout_width="1000dip" android:src="@drawable/line"
android:layout_height="5dip" android:id="@+id/imageButton18" android:layout_below="@+id/radioGroup3"
android:layout_alignParentLeft="true" android:layout_marginTop="29dp"></ImageButton>
        <ImageButton android:layout_width="5dip" android:src="@drawable/line"
android:layout_height="800dip" android:id="@+id/imageButton19" android:layout_alignParentTop="true"
android:layout_above="@+id/imageButton9" android:layout_toRightOf="@+id/imageButton9"></ImageButton>
        <TextView android:layout_width="wrap_content" android:text="二" android:textSize="30sp"
android:id="@+id/textView7" android:layout_height="wrap_content"
android:layout_below="@+id/imageButton14" android:layout_alignRight="@+id/textView6"></TextView>
        <TextView android:layout_width="wrap_content" android:text="调" android:textSize="30sp"
android:id="@+id/textView8" android:layout_height="wrap_content"
android:layout_below="@+id/imageButton15" android:layout_alignRight="@+id/textView7"></TextView>
        <TextView android:layout_width="wrap_content" android:text="光" android:textSize="30sp"
android:id="@+id/textView9" android:layout_height="wrap_content"
android:layout_below="@+id/imageButton16" android:layout_alignRight="@+id/textView8"></TextView>
        <TextView android:layout_height="wrap_content" android:text="界" android:textSize="30sp"
android:id="@+id/textView10" android:layout_width="wrap_content" android:layout_above="@+id/imageButton17"
android:layout_alignRight="@+id/textView9"></TextView>
        <TextView android:layout_height="wrap_content" android:text="面" android:textSize="30sp"
android:id="@+id/textView11" android:layout_width="wrap_content"
```

```
android:layout_below="@+id/imageButton17" android:layout_alignRight="@+id/textView10"></TextView>
            <TextView android:layout_height="wrap_content" android:text="第" android:textSize="30sp"
android:id="@+id/textView6" android:layout_width="wrap_content" android:layout_above="@+id/textView7"
android:layout_toRightOf="@+id/imageButton19"></TextView>


     </RelativeLayout>

</LinearLayout>
```

11.2　实现程序文件

知识点讲解：光盘:视频\视频讲解\第 11 章\实现程序文件.avi

在上一节的内容中，已经介绍了布局文件的设计过程，本节将详细讲解本系统实例的具体 Activity 程序文件的实现过程。

11.2.1　主 Activity

编写文件 Main.java，功能是响应用户按键处理事件，根据用户触摸的选项来到对应的模式，通过动画效果过渡来到 HOME 界面。文件 Main.java 的具体实现代码如下所示。

```
public class Main extends ActivityGroup implements OnGestureListener,OnTouchListener {
      //声明 ViewFlipper 对象
      private ViewFlipper m_ViewFlipper;
      //声明 GestureDetector 对象
      private GestureDetector m_GestureDetector;
      //声明 LocalActivityManager 对象
      private LocalActivityManager m_ActivityManager;
      private static int FLING_MIN_DISTANCE = 100;
      private static int FLING_MIN_VELOCITY = 200;
      //定义自定义图片加文字按钮 ImageButton 对象
      //private ImageButton mButton1;

      //单选按键部分 1
      private String[ ] areas = new String[ ]{"一般模式", "会议模式", "视频模式","迎接模式", "返回"};
      private RadioOnClick radioOnClick = new RadioOnClick(4);
      @SuppressWarnings("unused")
      private ListView RadioListView;
     private ImageButton imagebtn,imagebutton;
     OutputStream tmpOut = null;
     Timer timer=new Timer();
     @Override
     public void onCreate(Bundle savedInstanceState)
     {
          super.onCreate(savedInstanceState);
          requestWindowFeature(Window.FEATURE_NO_TITLE);
          //设置内容视图
          setContentView(R.layout.main);
```

```
            getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
            imagebutton=(ImageButton)findViewById(R.id.imageButton1);
            imagebutton.setOnClickListener(new ImageButton.OnClickListener()
            {
              @Override
                  public void onClick(View v)
              {
                     /*Intent intent=new Intent();
                     intent.setClass(Main.this, stand.class);
                     startActivity(intent);
                     Main.this.finish();
                     overridePendingTransition(R.anim.zoomin,R.anim.zoomout);*/
                     LayoutInflater inflater = getLayoutInflater();
                        View layout = inflater.inflate(R.layout.lightstandard,
                        (ViewGroup) findViewById(R.id.stand));

                         new AlertDialog.Builder(Main.this)
                         .setTitle("Ihe nationl standard of illumination")
                         .setView(layout)
                         .setPositiveButton("Return", null)
                         //.setNegativeButton("取消", null)
                         .show();
              }
            }
                     );
            //单选按键部分 2
            imagebtn=(ImageButton)findViewById(R.id.imageButton2);
            imagebtn.setOnClickListener(new RadioClickListener());
            //构建 ViewFlipper 对象
        m_ViewFlipper = (ViewFlipper) findViewById(R.id.fliper);
        //获取 Activity 消息
        m_ActivityManager = getLocalActivityManager();
        //注册一个用于手势识别的类
        m_GestureDetector = new GestureDetector(this);
        //添加视图，指定每个视图对应的 Activity
        m_ViewFlipper.addView((m_ActivityManager.startActivity("", new Intent(Main.this,Firstpage.class)).
getDecorView()),0);
        m_ViewFlipper.addView((m_ActivityManager.startActivity("", new Intent(Main.this,Secondpage.class)).
getDecorView()),1);
        //给 ViewFlipper 设置一个 Listener
        m_ViewFlipper.setOnTouchListener(this);
        //默认为正在播放页面并设置图标
        //设置相应元素索引显示的子视图
        m_ViewFlipper.setDisplayedChild(0);
        //允许长按住 ViewFlipper，这样才能识别拖动等手势
        m_ViewFlipper.setLongClickable(true);
        //监听
             /** Called when the activity is first created */
            /*取得  Button 对象*/
        //返回按钮按键事件
```

```
    /* btnbac = (ImageButton) findViewById(R.id.btnback);
     btnbac.setOnClickListener(new ImageButton.OnClickListener(){
      @Override
      public void onClick(View v)
      {
         // TODO Auto-generated method stub
        Intent intent=new Intent();
        intent.setClass(Main.this, home.class);
        startActivity(intent);
        Main.this.finish();
        overridePendingTransition(R.anim.zoomin,R.anim.zoomout);
        /*Intent intent = new Intent();
        intent.setClass(Main.this, home.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);   //注意本行的 Flag 设置
        startActivity(intent);

      } });*/}
  @Override
   public boolean onKeyDown(int keyCode, KeyEvent event) {
        //按下键盘上的返回按钮
        if(keyCode == KeyEvent.KEYCODE_BACK){
             new AlertDialog.Builder(this)
                  //.setIcon(R.drawable.services)
            .setTitle(R.string.app_about)
              /*设置弹出窗口的图式*/
            // .setIcon(R.drawable.hot)
              /*设置弹出窗口的信息*/
              .setMessage(R.string.app_about_msg)
              .setPositiveButton(R.string.str_ok,
              new DialogInterface.OnClickListener()
           {
            public void onClick(DialogInterface dialoginterface, int i)
            {
              finish();/*关闭窗口*/
            }
           }
           )
              /*设置弹出窗口的返回事件*/
            .setNegativeButton(R.string.str_no,
               new DialogInterface.OnClickListener()
              {
            public void onClick(DialogInterface dialoginterface, int i)
           {
           }
              })
           .show();
             //setResult(11,this.getIntent());
             return true;
        }else{
             return super.onKeyDown(keyCode, event);
        }
```

```
        }
        @Override
        protected void onDestroy() {
                super.onDestroy();


                android.os.Process.killProcess(android.os.Process.myPid());
                System.exit(0);
                overridePendingTransition(R.anim.zoomin,R.anim.zoomout);



        }
  /**
         * 定义从右侧进入的动画效果
         * @return
         */
        public Animation inFromRightAnimation()
        {
         Animation inFromRight = new TranslateAnimation(
                        Animation.RELATIVE_TO_PARENT, +1.0f,
                        Animation.RELATIVE_TO_PARENT, 0.0f,
                        Animation.RELATIVE_TO_PARENT, 0.0f,
                        Animation.RELATIVE_TO_PARENT, 0.0f);
         inFromRight.setDuration(500);
         inFromRight.setInterpolator(new AccelerateInterpolator());
         return inFromRight;
        }
        /**
         * 定义从左侧退出的动画效果
         * @return
         */
        public Animation outToLeftAnimation()
        {
         Animation outtoLeft = new TranslateAnimation(
                        Animation.RELATIVE_TO_PARENT, 0.0f,
                        Animation.RELATIVE_TO_PARENT, -1.0f,
                        Animation.RELATIVE_TO_PARENT, 0.0f,
                        Animation.RELATIVE_TO_PARENT, 0.0f);
         outtoLeft.setDuration(500);
         outtoLeft.setInterpolator(new AccelerateInterpolator());
         return outtoLeft;
        }
        /**
         * 定义从左侧进入的动画效果
         * @return
         */
        public Animation inFromLeftAnimation()
        {
         Animation inFromLeft = new TranslateAnimation(
                        Animation.RELATIVE_TO_PARENT, -1.0f,
```

```
                Animation.RELATIVE_TO_PARENT, 0.0f,
                Animation.RELATIVE_TO_PARENT, 0.0f,
                Animation.RELATIVE_TO_PARENT, 0.0f);
 inFromLeft.setDuration(500);
 inFromLeft.setInterpolator(new AccelerateInterpolator());
 return inFromLeft;
}
/**
 * 定义从右侧退出时的动画效果
 * @return
 */
public Animation outToRightAnimation()
{
 Animation outtoRight = new TranslateAnimation(
                Animation.RELATIVE_TO_PARENT, 0.0f,
                Animation.RELATIVE_TO_PARENT, +1.0f,
                Animation.RELATIVE_TO_PARENT, 0.0f,
                Animation.RELATIVE_TO_PARENT, 0.0f);
 outtoRight.setDuration(500);
 outtoRight.setInterpolator(new AccelerateInterpolator());
 return outtoRight;
}

 @Override
 public boolean onDown(MotionEvent e) {
        return false;
 }
 @Override
 public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
                float velocityY) {
 //当向左侧滑动时
 if(e1.getX()-e2.getX()>FLING_MIN_DISTANCE && Math.abs(velocityX)>FLING_MIN_VELOCITY)
 {
        //设置 View 进入屏幕时使用的动画
        m_ViewFlipper.setInAnimation(inFromRightAnimation());
        //设置 View 退出屏幕时使用的动画
        m_ViewFlipper.setOutAnimation(outToLeftAnimation());
        //下一个页面
        m_ViewFlipper.showNext();
        //获取相应元素索引显示的子视图
 }
 //当向右侧滑动时
 else if(e2.getX()-e1.getX()>FLING_MIN_DISTANCE && Math.abs(velocityX)>FLING_MIN_VELOCITY)
        {
                //设置 View 进入屏幕时使用的动画
                m_ViewFlipper.setInAnimation(inFromLeftAnimation());
                //设置 View 退出屏幕时使用的动画
                m_ViewFlipper.setOutAnimation(outToRightAnimation());
                //上一个页面
                m_ViewFlipper.showPrevious();
                //获取相应元素索引显示的子视图
```

```
            }
            return false;
    }
    @Override
    public void onLongPress(MotionEvent e) {

    }
    @Override
    public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX,
                float distanceY) {

            return false;
    }
    @Override
    public void onShowPress(MotionEvent e) {

    }
    @Override
    public boolean onSingleTapUp(MotionEvent e) {
            return false;
    }
    @Override
    public boolean onTouch(View v, MotionEvent event) {
            //一定要将触屏事件交给手势识别类去处理（自己处理会很麻烦）
            return m_GestureDetector.onTouchEvent(event);
    }

   //单选按键部分 3
   class RadioClickListener implements OnClickListener {
      @Override
      public void onClick(View v) {
       AlertDialog ad =new AlertDialog.Builder(Main.this).setTitle("选择模式")
       .setSingleChoiceItems(areas,radioOnClick.getIndex(),radioOnClick).create();
       RadioListView=ad.getListView();
       ad.show();
   }
   }

   /**
    * 单击单选按钮事件
    *
    */
   class RadioOnClick implements DialogInterface.OnClickListener{
 private int index;

 public RadioOnClick(int index){
  this.index = index;
 }
 public void setIndex(int index){
  this.index=index;
 }
```

```
public int getIndex(){
  return index;
}

public void onClick(DialogInterface dialog, int whichButton){
    setIndex(whichButton);
  // Toast.makeText(Main.this, "您选择了：" + areas[index], Toast.LENGTH_LONG).show();
    switch (index)
    {
    //一般模式下
        case 0:

        {
            //第一个子节点的亮度
            send(0x01,0x5a);

            //第二个子节点的亮度
            send(0x02,0x5a);

            //第三个子节点的亮度
            send(0x03,0x5a);

            //第四个子节点的亮度
            send(0x04,0x5a);
     Toast.makeText(Main.this, "您选择了：" + areas[index], Toast.LENGTH_LONG).show();
       dialog.dismiss();
    }
    break;


  //会议模式下
        case 1:
        {
            //第一路子节点
            send(0x01,0x23);

            //第二路子节点
            send(0x02,0x23);

            //第三路子节点
            send(0x03,0x23);

            //第四路子节点
            send(0x04,0x23);
            Toast.makeText(Main.this, "您选择了：" + areas[index], Toast.LENGTH_LONG).show();
            dialog.dismiss();
        }    break;


    //视频模式下
        case 2:
```

```
    {
        //第一路子节点
        send(0x01,0xc8);

        //第二路子节点
        send(0x02,0xc8);

        //第三路子节点
        send(0x03,0xc8);

        //第四路子节点
        send(0x04,0xc8);

        Toast.makeText(Main.this, "您选择了：" + areas[index], Toast.LENGTH_LONG).show();
        dialog.dismiss();
  }    break;


  //迎接模式下
    case 3:
    {
        //第一路子节点
        send(0x01,0x23);

        //第二路子节点
        TimerTask timerTask = new TimerTask() {
              @Override
              public void run()
              {
                          //实现信息发送
                          send(0x02,0x23);
              }
    };
    timer.schedule(timerTask, 1000 * 3);
        //第三路子节点
     TimerTask timerTask1 = new TimerTask() {
         @Override
         public void run() {

                         //实现信息发送
           send(0x03,0x23);

         }
    };
          timer.schedule(timerTask1, 1000 * 6); //2 秒后执行

          //第四路子节点
     TimerTask timerTask2 = new TimerTask() {
           @Override
           public void run() {
                        //实现信息发送
```

```
                    send(0x04,0x23);

                  }};
                  timer.schedule(timerTask2, 1000 * 9);
           Toast.makeText(Main.this, "您选择了：" + areas[index], Toast.LENGTH_LONG).show();
            dialog.dismiss();
      }    break;

          //返回
          case 4:

          {//Toast.makeText(Main.this, "您选择了：" + areas[index], Toast.LENGTH_LONG).show();
            dialog.dismiss();   }
  }


}

  private void   send(int Room,int Grade){

          try {

                  //String strpass="@@UP...";
                 byte[ ] byteone=new byte[8];//=strpass.getBytes("US-ASCII");
                      byteone[0]=(byte)0xf5;
                      byteone[1]=(byte)0x5f;
                      byteone[2]=(byte)0x00;
                      byteone[3]=(byte)Room;
                      byteone[4]=(byte)0x03;
                      byteone[5]=(byte)0x00;
                      byteone[6]=(byte)Grade;
                      byteone[7]=(byte)0x06;

                    //byte[ ] bytekai=strpass.getBytes("US-ASCII");
             tmpOut = BluetoothMain.btSocket.getOutputStream();
             tmpOut.write(byteone);}
         catch (IOException e) {
             Log.e("BluetoothReadService", "temp sockets not created", e);
         }
           }
  }}
```

11.2.2　监听单击事件

编写文件 home.java，功能是监听用户触摸单击的选项来执行对应的处理程序，来到对应的 Activity 界面。文件 home.java 的具体实现代码如下所示。

```
public class home extends Activity {
      private ImageButton enterr,enterp,enterc,back;
     /** Called when the activity is first created */
     @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.home);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,  WindowManager.Layout
Params.FLAG_FULLSCREEN);
        enterr=(ImageButton) findViewById (R.id.Enterr);
        enterp=(ImageButton) findViewById (R.id.Enterp);
        enterc=(ImageButton) findViewById (R.id.Enterc);
        back=(ImageButton) findViewById (R.id.back);
        enterr.setOnClickListener(new Button.OnClickListener()
        {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Intent intent = new Intent();
            intent.setClass(home.this, BluetoothMain.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);  //注意本行的 Flag 设置
            startActivity(intent);
             //home.this.finish();
            overridePendingTransition(R.anim.zoomin,R.anim.zoomout);


            }

        }
            );

        enterp.setOnClickListener(new Button.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                //TODO Auto-generated method stub
                Intent intent=new Intent();
            intent.setClass(home.this, product.class);
            startActivity(intent);
            home.this.finish();
            overridePendingTransition(R.anim.zoomin,R.anim.zoomout);

        }}
            );

        enterc.setOnClickListener(new Button.OnClickListener()

        {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Intent intent=new Intent();
```

```
                intent.setClass(home.this, company.class);
                startActivity(intent);
                home.this.finish();
                overridePendingTransition(R.anim.zoomin,R.anim.zoomout);

        }}
                );

        back.setOnClickListener(new Button.OnClickListener()
        {

                @Override
                public void onClick(View v) {

                    AlertDialog.Builder alertbBuilder=new AlertDialog.Builder(home.this);
                        //.setIcon(R.drawable.services)
                        alertbBuilder.setTitle(R.string.app_about)
                    /*设置弹出窗口的图式*/
                    //.setIcon(R.drawable.hot)
                    /*设置弹出窗口的信息*/
                    .setMessage(R.string.app_about_msg)
                    .setPositiveButton(R.string.str_ok,
                    new DialogInterface.OnClickListener()
                  {
                   public void onClick(DialogInterface dialoginterface, int i)
                   {
                    finish();/*关闭窗口*/
                   }
                  }
                  )
                    /*设置弹出窗口的返回事件*/
                    .setNegativeButton(R.string.str_no,
                     new DialogInterface.OnClickListener()
                    {
                   public void onClick(DialogInterface dialoginterface, int i)
                  {
                  }
                    })
                  .show();

                }
    });}
}
```

11.2.3 设置系统的蓝牙参数

编写文件 BluetoothMain.java，功能是根据用户的设置选项来设置系统的蓝牙参数，实现控制本系统蓝牙设备的功能。文件 BluetoothMain.java 的具体实现代码如下所示。

```
public class BluetoothMain extends Activity {
     static final String SPP_UUID = "00001101-0000-1000-8000-00805F9B34FB";
```

```
Button btnSearch, btnDis, btnExit;
ToggleButton tbtnSwitch;
ListView lvBTDevices;
ArrayAdapter<String> adtDevices;
List<String> lstDevices = new ArrayList<String>();

BluetoothAdapter btAdapt;
public static BluetoothSocket btSocket;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.bluetooth);
    //Button 设置
    btnSearch = (Button) this.findViewById(R.id.btnSearch);
    btnSearch.setOnClickListener(new ClickEvent());
    btnExit = (Button) this.findViewById(R.id.btnExit);
    btnExit.setOnClickListener(new ClickEvent());
    btnDis = (Button) this.findViewById(R.id.btnDis);
    btnDis.setOnClickListener(new ClickEvent());

    //ToogleButton 设置
    tbtnSwitch = (ToggleButton) this.findViewById(R.id.tbtnSwitch);
    tbtnSwitch.setOnClickListener(new ClickEvent());

    // ListView 及其数据源适配器
    lvBTDevices = (ListView) this.findViewById(R.id.lvDevices);
    adtDevices = new ArrayAdapter<String>(BluetoothMain.this,
            android.R.layout.simple_list_item_1, lstDevices);
    lvBTDevices.setAdapter(adtDevices);
    //lvBTDevices.setOnItemClickListener(new ItemClickEvent());

    // 初始化本机蓝牙功能，读取蓝牙状态并显示
    btAdapt = BluetoothAdapter.getDefaultAdapter();
   if (btAdapt.getState() == BluetoothAdapter.STATE_OFF)
        tbtnSwitch.setChecked(false);
    else if (btAdapt.getState() == BluetoothAdapter.STATE_ON)
        tbtnSwitch.setChecked(true);

    // 注册 Receiver 来获取蓝牙设备相关的结果
    IntentFilter intent = new IntentFilter();
    intent.addAction(BluetoothDevice.ACTION_FOUND);// 用 BroadcastReceiver 获取搜索结果
    intent.addAction(BluetoothDevice.ACTION_BOND_STATE_CHANGED);
    intent.addAction(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED);
    intent.addAction(BluetoothAdapter.ACTION_STATE_CHANGED);
    registerReceiver(searchDevices, intent);

    if (btAdapt.getState() == BluetoothAdapter.STATE_OFF) {//如果蓝牙还没开启
        Toast.makeText(BluetoothMain.this, "Bluetooth is openning，Just a minute , please", 1000).show();
        btAdapt.enable();
```

```
            tbtnSwitch.setChecked(true);
        }
        setTitle("The Bluetooth address：" + btAdapt.getAddress());
        lstDevices.clear();
        btAdapt.startDiscovery();

    }



    private BroadcastReceiver searchDevices = new BroadcastReceiver() {

        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            Bundle b = intent.getExtras();
            Object[ ] lstName = b.keySet().toArray();

            //显示所有收到的消息及其细节
            for (int i = 0; i < lstName.length; i++) {
                String keyName = lstName[i].toString();
                Log.e(keyName, String.valueOf(b.get(keyName)));
            }

            //搜索设备时，取得设备的 MAC 地址
            if (BluetoothDevice.ACTION_FOUND.equals(action)) {
                BluetoothDevice device = intent
                        .getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                String str= device.getName() + "|" + device.getAddress();
                String str1 = device.getAddress();
            //if(str1.equals("00:11:08:01:06:78"))   //用来判断是否是所需要的蓝牙
            //{

                if (lstDevices.indexOf(str) == -1)//防止重复添加
                    lstDevices.add(str); //获取设备名称和 MAC 地址
                adtDevices.notifyDataSetChanged();
                /*   添加判断，如果为已配对或者已存 MAC 地址的设备
                 *   自动进行连接，并跳转到另一个 Activity
                 * */
                if(true){     //未添加条件
                    btAdapt.cancelDiscovery();
                    //String str = lstDevices.get();
                    //String[ ] values = str.split("\\|");
                    //String address=values[1];
                    //Log.e("address",values[1]);
                    UUID uuid = UUID.fromString(SPP_UUID);
                    BluetoothDevice btDev = btAdapt.getRemoteDevice(device.getAddress());
                    try {
                        btSocket = btDev
                                .createRfcommSocketToServiceRecord(uuid);
                        btSocket.connect();
```

```
                                        Intent intent1 = new Intent();
                                        intent1.setClass(BluetoothMain.this, Main.class);
                                        startActivity(intent1);
                                        overridePendingTransition(R.anim.zoomin,R.anim.zoomout);
                                } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                        }


                //}
                }
        }
};

@Override
protected void onDestroy() {
        this.unregisterReceiver(searchDevices);
        super.onDestroy();
        android.os.Process.killProcess(android.os.Process.myPid());
}
class ClickEvent implements View.OnClickListener {
        @Override
        public void onClick(View v) {
                if (v == btnSearch)//搜索蓝牙设备，在 BroadcastReceiver 中显示结果
                {

                        if (btAdapt.getState() == BluetoothAdapter.STATE_OFF) {//如果蓝牙还没开启
                                Toast.makeText(BluetoothMain.this, "请先打开蓝牙", 1000).show();
                                return;
                        }

                        setTitle("本机蓝牙地址：" + btAdapt.getAddress());
                        lstDevices.clear();
                        btAdapt.startDiscovery();
                } else if (v == tbtnSwitch) {//本机蓝牙启动/关闭
                        if (btAdapt.getState() == BluetoothAdapter.STATE_OFF){
                                btAdapt.enable();
                                tbtnSwitch.setChecked(true);
                        }

                        else if (btAdapt.getState() == BluetoothAdapter.STATE_ON){
                                btAdapt.disable();
                                tbtnSwitch.setChecked(false);
                        }
                } else if (v == btnDis)//本机可以被搜索
                {
                        Intent discoverableIntent = new Intent(
                                        BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
                        discoverableIntent.putExtra(
```

```
                    BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, 300);
                startActivity(discoverableIntent);
            } else if (v == btnExit) {
                try {
                    if (btSocket != null)
                        btSocket.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
             if (btAdapt.getState() == BluetoothAdapter.STATE_ON){
                    btAdapt.disable();
             }
                BluetoothMain.this.finish();
            }
        }
    }
```

11.2.4　控制第一路到第四路光线的亮度

编写文件 Firstpage.java，功能是监听用户单选按钮列表中的选择值，根据这个值来控制第一路到第四路光线的亮度。文件 Firstpage.java 的具体实现代码如下所示。

```
public class Firstpage extends Activity {
    private ImageButton imagebutton;
    private ImageButton ibutton1_o,ibutton1_c,ibutton2_o,ibutton2_c,ibutton3_o,
    ibutton3_c,ibutton4_o,ibutton4_c,btnallop,btnallcl;
    RadioGroup radiogroup0,radiogroup1,radiogroup2,radiogroup3;
    RadioButton radio1,radio2,radio3,radio4,radio5,radio6;
    RadioButton radio11,radio12,radio13,radio14,radio15,radio16;
    RadioButton radio21,radio22,radio23,radio24,radio25,radio26;
    RadioButton radio31,radio32,radio33,radio34,radio35,radio36;
    OutputStream tmpOut = null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.first);
        //全开按钮
        btnallop=(ImageButton) findViewById(R.id.btnopen);
        btnallop.setOnClickListener(new Button.OnClickListener(){
            @Override
            public void onClick(View v)
            {
             //第一路灯亮
             send(0x01,0x23);


             //第二路灯亮
             send(0x02,0x23);

             //第三路灯亮
             send(0x03,0x23);
```

```
            //第四路灯亮
            send(0x04,0x23);


        }
        });

    //全关按钮

    btnallcl=(ImageButton) findViewById(R.id.btnclose);
    btnallcl.setOnClickListener(new Button.OnClickListener(){
            @Override
            public void onClick(View v)
            {
             //第一路灯亮
             send(0x01,0xff);

             //第二路灯亮
             send(0x02,0xff);

             send(0x03,0xff);

             send(0x04,0xff);

            }
            });
    //产品介绍按钮
    imagebutton=(ImageButton) findViewById(R.id.imageButton1);
    imagebutton.setOnClickListener(new Button.OnClickListener(){
            @Override
            public void onClick(View v)
            {
              LayoutInflater inflater = getLayoutInflater();
             View layout = inflater.inflate(R.layout.dialog,
             (ViewGroup) findViewById(R.id.dialog));

              new AlertDialog.Builder(Firstpage.this)
              .setTitle("Introduce")
              .setView(layout)
              .setPositiveButton("Return", null)
              //.setNegativeButton("取消", null)
              .show();
            }
      });

      //第一子节点的占空比
      radiogroup0=(RadioGroup)findViewById(R.id.radioGroup2);
         radio1=(RadioButton)findViewById(R.id.radio11);
         radio2=(RadioButton)findViewById(R.id.radio12);
         radio3=(RadioButton)findViewById(R.id.radio13);
```

```
        radio4=(RadioButton)findViewById(R.id.radio14);
        radio5=(RadioButton)findViewById(R.id.radio15);
        radio6=(RadioButton)findViewById(R.id.radio16);
        radiogroup0.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        // TODO Auto-generated method stub

        switch (checkedId)
        {
        case R.id.radio11:
                /*DisplayToast("灯光亮度 0%！ ");*/
             Toast.makeText(Firstpage.this, "第一路灯光亮度为 0%！ ",
                        Toast.LENGTH_SHORT).show();
        {
             send(0x01,0xff);

        }
             break;
        case R.id.radio12:
               // DisplayToast("灯光亮度 20%！ ");
             Toast.makeText(Firstpage.this, "第一路灯光亮度为 20%！ ",
                             Toast.LENGTH_SHORT).show();
        {
             send(0x01,0xc8);

        }
             break;
        case R.id.radio13:
               //DisplayToast("灯光亮度 40%！ ");
             Toast.makeText(Firstpage.this, "第一路灯光亮度为 40%！ ",
                             Toast.LENGTH_SHORT).show();
        {
             send(0x01,0x91);

        }
             break;
        case R.id.radio14:
                //DisplayToast("灯光亮度 60%！ ");
             Toast.makeText(Firstpage.this, "第一路灯光亮度为 60%！ ",
                             Toast.LENGTH_SHORT).show();
        {
             send(0x01,0x5a);

        }
             break;
        case R.id.radio15:
               // DisplayToast("灯光亮度 80%！ ");
             Toast.makeText(Firstpage.this, "第一路灯光亮度为 80%！ ",
                             Toast.LENGTH_SHORT).show();
```

```
            {
                send(0x01,0x23);

            }
                break;
            default:
                // DisplayToast("灯光亮度 100%！");
                Toast.makeText(Firstpage.this, "第一路灯光亮度为 100%！",
                                Toast.LENGTH_SHORT).show();
            {
                send(0x01,0x00);

            }
                break;
            }

        }
    });
    //第二子节点的占空比
    radiogroup1=(RadioGroup)findViewById(R.id.radioGroup3);
    radio11=(RadioButton)findViewById(R.id.radio21);
    radio12=(RadioButton)findViewById(R.id.radio22);
    radio13=(RadioButton)findViewById(R.id.radio23);
    radio14=(RadioButton)findViewById(R.id.radio24);
    radio15=(RadioButton)findViewById(R.id.radio25);
    radio16=(RadioButton)findViewById(R.id.radio26);
    radiogroup1.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

        @Override
        public void onCheckedChanged(RadioGroup group, int checkedId) {
            // TODO Auto-generated method stub

            switch (checkedId)
            {
            case R.id.radio21:
                /*DisplayToast("灯光亮度 0%！");*/
                Toast.makeText(Firstpage.this, "第二路灯光亮度为 0%！",
                            Toast.LENGTH_SHORT).show();
            {
                send(0x02,0xff);

            }
                break;
            case R.id.radio22:
                // DisplayToast("灯光亮度 20%！");
                Toast.makeText(Firstpage.this, "第二路灯光亮度为 20%！",
                                Toast.LENGTH_SHORT).show();
            {
                send(0x02,0xc8);

            }
```

```
                break;
            case R.id.radio23:
                  //DisplayToast("灯光亮度 40%！");
                 Toast.makeText(Firstpage.this, "第二路灯光亮度为 40%！",
                                   Toast.LENGTH_SHORT).show();
            {
                 send(0x02,0x91);

            }
                break;
            case R.id.radio24:
                   //DisplayToast("灯光亮度 60%！");
                 Toast.makeText(Firstpage.this, "第二路灯光亮度为 60%！",
                                   Toast.LENGTH_SHORT).show();
            {
                 send(0x02,0x5a);

            }
                break;
            case R.id.radio25:
                  // DisplayToast("灯光亮度 80%！");
                 Toast.makeText(Firstpage.this, "第二路灯光亮度为 80%！",
                                   Toast.LENGTH_SHORT).show();
            {
                 send(0x02,0x23);

            }
                break;
            default:
                  // DisplayToast("灯光亮度 100%！");
                 Toast.makeText(Firstpage.this, "第二路灯光亮度为 100%！",
                                   Toast.LENGTH_SHORT).show();
            {
                 send(0x02,0x00);

            }
                break;
            }

        }
    });

    //第三子节点的占空比
    radiogroup2=(RadioGroup)findViewById(R.id.radioGroup4);
    radio21=(RadioButton)findViewById(R.id.radio31);
    radio22=(RadioButton)findViewById(R.id.radio32);
    radio23=(RadioButton)findViewById(R.id.radio33);
    radio24=(RadioButton)findViewById(R.id.radio34);
    radio25=(RadioButton)findViewById(R.id.radio35);
    radio26=(RadioButton)findViewById(R.id.radio36);
    radiogroup2.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
```

```
@Override
public void onCheckedChanged(RadioGroup group, int checkedId) {
    // TODO Auto-generated method stub

  switch (checkedId)
  {
  case R.id.radio31:
         /*DisplayToast("灯光亮度 0%！");*/
       Toast.makeText(Firstpage.this, "第三路灯光亮度为 0%！",
                 Toast.LENGTH_SHORT).show();
  {
       send(0x03,0xff);

  }
      break;
  case R.id.radio32:
        // DisplayToast("灯光亮度 20%！");
       Toast.makeText(Firstpage.this, "第三路灯光亮度为 20%！",
                      Toast.LENGTH_SHORT).show();
  {
       send(0x03,0xc8);

  }
      break;
  case R.id.radio33:
        //DisplayToast("灯光亮度 40%！");
       Toast.makeText(Firstpage.this, "第三路灯光亮度为 40%！",
                      Toast.LENGTH_SHORT).show();
  {
       send(0x03,0x91);

  }
      break;
  case R.id.radio34:
         //DisplayToast("灯光亮度 60%！");
       Toast.makeText(Firstpage.this, "第三路灯光亮度为 60%！",
                      Toast.LENGTH_SHORT).show();
  {
       send(0x03,0x5a);

  }
      break;
  case R.id.radio35:
        // DisplayToast("灯光亮度 80%！");
       Toast.makeText(Firstpage.this, "第三路灯光亮度为 80%！",
                      Toast.LENGTH_SHORT).show();
  {
       send(0x03,0x23);

  }
```

```
                break;
            default:
                  // DisplayToast("灯光亮度 100%！");
                 Toast.makeText(Firstpage.this, "第三路灯光亮度为 100%！",
                                        Toast.LENGTH_SHORT).show();
            {
                 send(0x03,0x00);

            }
                break;
            }

        }
    });    //第四路

    //第四子节点的占空比
    radiogroup3=(RadioGroup)findViewById(R.id.radioGroup5);
    radio31=(RadioButton)findViewById(R.id.radio41);
    radio32=(RadioButton)findViewById(R.id.radio42);
    radio33=(RadioButton)findViewById(R.id.radio43);
    radio34=(RadioButton)findViewById(R.id.radio44);
    radio35=(RadioButton)findViewById(R.id.radio45);
    radio36=(RadioButton)findViewById(R.id.radio46);
    radiogroup3.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

        @Override
        public void onCheckedChanged(RadioGroup group, int checkedId) {
            // TODO Auto-generated method stub

            switch (checkedId)
            {
            case R.id.radio41:
                   /*DisplayToast("灯光亮度 0%！");*/
                 Toast.makeText(Firstpage.this, "第四路灯光亮度为 0%！",
                                  Toast.LENGTH_SHORT).show();
            {
                 send(0x04,0xff);

            }
                break;
            case R.id.radio42:
                  // DisplayToast("灯光亮度 20%！");
                 Toast.makeText(Firstpage.this, "第四路灯光亮度为 20%！",
                                        Toast.LENGTH_SHORT).show();
            {
                 send(0x04,0xc8);

            }
                break;
            case R.id.radio43:
                  //DisplayToast("灯光亮度 40%！");
```

```
                Toast.makeText(Firstpage.this, "第四路灯光亮度为 40%！",
                                Toast.LENGTH_SHORT).show();
            {
                send(0x04,0x91);

            }
                break;
            case R.id.radio44:
                    //DisplayToast("灯光亮度 60%！");
                Toast.makeText(Firstpage.this, "第四路灯光亮度为 60%！",
                                Toast.LENGTH_SHORT).show();
            {
                send(0x04,0x5a);

            }
                break;
            case R.id.radio45:
                    // DisplayToast("灯光亮度 80%！");
                Toast.makeText(Firstpage.this, "第四路灯光亮度为 80%！",
                                Toast.LENGTH_SHORT).show();
            {
                send(0x04,0x23);

            }
                break;
            default:
                    // DisplayToast("灯光亮度 100%！");
                Toast.makeText(Firstpage.this, "第四路灯光亮度为 100%！",
                                Toast.LENGTH_SHORT).show();
            {
                send(0x04,0x00);

            }
                break;
            }

        }
    });



    ibutton1_o=(ImageButton) findViewById(R.id.imageButton2);
    ibutton1_o.setOnClickListener(new ClickEventKey());
    ibutton1_c=(ImageButton) findViewById(R.id.imageButton3);
    ibutton1_c.setOnClickListener(new ClickEventKey());
    ibutton2_o=(ImageButton) findViewById(R.id.imageButton4);
    ibutton2_o.setOnClickListener(new ClickEventKey());
    ibutton2_c=(ImageButton) findViewById(R.id.imageButton5);
    ibutton2_c.setOnClickListener(new ClickEventKey());
    ibutton3_o=(ImageButton) findViewById(R.id.imageButton6);
    ibutton3_o.setOnClickListener(new ClickEventKey());
```

```
        ibutton3_c=(ImageButton) findViewById(R.id.imageButton7);
        ibutton3_c.setOnClickListener(new ClickEventKey());
        ibutton4_o=(ImageButton) findViewById(R.id.imageButton8);
        ibutton4_o.setOnClickListener(new ClickEventKey());
        ibutton4_c=(ImageButton) findViewById(R.id.imageButton9);
        ibutton4_c.setOnClickListener(new ClickEventKey());
 }



class ClickEventKey implements View.OnClickListener {
 @Override
 public void onClick(View v) {
//      if (v == ibutton1_o)
        switch (v.getId())

        {
        case R.id.imageButton2:
        {
            send(0x01,0x23);
        }

        break;
        case R.id.imageButton3:

        {
            send(0x01,0xff);
        }

        break;
        case R.id.imageButton4:
        {
            send(0x02,0x23);
        }

        break;
        case R.id.imageButton5:
        {
            send(0x02,0xff);
        }

        break;
            case R.id.imageButton6:
        {
           send(0x03,0x23);
        }

        break;
            case R.id.imageButton7:
        {
```

```
                    send(0x03,0xff);
            }

            break;
                    case R.id.imageButton8:
            {
                    send(0x04,0x23);
            }

            break;
                    case R.id.imageButton9:
            {
                    send(0x04,0xff);
            }

            break;
    }
            }
    }
private void    send(int Room,int Grade){

            try {

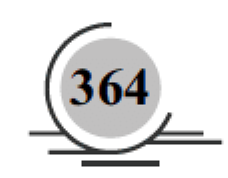
                        //String strpass="@@UP...";
                        byte[ ] byteone=new byte[8];//=strpass.getBytes("US-ASCII");
                              byteone[0]=(byte)0xf5;
                              byteone[1]=(byte)0x5f;
                              byteone[2]=(byte)0x00;
                              byteone[3]=(byte)Room;
                              byteone[4]=(byte)0x03;
                              byteone[5]=(byte)0x00;
                              byteone[6]=(byte)Grade;
                              byteone[7]=(byte)0x06;

                          //byte[ ] bytekai=strpass.getBytes("US-ASCII");
                    tmpOut = BluetoothMain.btSocket.getOutputStream();
                    tmpOut.write(byteone);}
             catch (IOException e) {
                    Log.e("BluetoothReadService", "temp sockets not created", e);
            }
            }
    }
```

11.2.5　控制第五路到第八路光线的亮度

编写文件 Secondpage.java，功能是监听用户单选按钮列表中的选择值，根据这个值来控制第五路到第八路光线的亮度。文件 Secondpage.java 的具体实现代码如下所示。

```
public class Secondpage extends Activity {
        private ImageButton imagebutton;
        private ImageButton ibutton5_o,ibutton5_c,ibutton6_o,ibutton6_c,ibutton7_o,
```



```
ibutton7_c,ibutton8_o,ibutton8_c;
RadioGroup radiogroup0,radiogroup1,radiogroup2,radiogroup3;
RadioButton radio1,radio2,radio3,radio4,radio5,radio6;
RadioButton radio11,radio12,radio13,radio14,radio15,radio16;
RadioButton radio21,radio22,radio23,radio24,radio25,radio26;
RadioButton radio31,radio32,radio33,radio34,radio35,radio36;
OutputStream tmpOut = null;
@Override
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.second);
      imagebutton=(ImageButton) findViewById(R.id.imageButton1);
      imagebutton.setOnClickListener(new Button.OnClickListener(){
             @Override
             public void onClick(View v)
             {
              LayoutInflater inflater = getLayoutInflater();
              View layout = inflater.inflate(R.layout.dialog,
              (ViewGroup) findViewById(R.id.dialog));

               new AlertDialog.Builder(Secondpage.this)
               .setTitle("Introduce")
               .setView(layout)
               .setPositiveButton("Return", null)
               //.setNegativeButton("取消", null)
               .show();
               }
 });

      radiogroup0=(RadioGroup)findViewById(R.id.radioGroup2);
      radio1=(RadioButton)findViewById(R.id.radio11);
      radio2=(RadioButton)findViewById(R.id.radio12);
      radio3=(RadioButton)findViewById(R.id.radio13);
      radio4=(RadioButton)findViewById(R.id.radio14);
      radio5=(RadioButton)findViewById(R.id.radio15);
      radio6=(RadioButton)findViewById(R.id.radio16);
      radiogroup0.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

         @Override
         public void onCheckedChanged(RadioGroup group, int checkedId) {
              // TODO Auto-generated method stub

           switch (checkedId)
           {
           case R.id.radio11:
                   /*DisplayToast("灯光亮度 0%！");*/
                 Toast.makeText(Secondpage.this, "第五路灯光亮度为 0%！",
                              Toast.LENGTH_SHORT).show();
           {
                 send(0x05,0xff);
           }
```

```
                break;
            case R.id.radio12:
                  // DisplayToast("灯光亮度 20%！");
                 Toast.makeText(Secondpage.this, "第五路灯光亮度为 20%！",
                                        Toast.LENGTH_SHORT).show();
            {
                 send(0x05,0xc8);
            }
                break;
            case R.id.radio13:
                  //DisplayToast("灯光亮度 40%！");
                 Toast.makeText(Secondpage.this, "第五路灯光亮度为 40%！",
                                        Toast.LENGTH_SHORT).show();
            {
                 send(0x05,0x91);
            }
                break;
            case R.id.radio14:
                   //DisplayToast("灯光亮度 60%！");
                 Toast.makeText(Secondpage.this, "第五路灯光亮度为 60%！",
                                        Toast.LENGTH_SHORT).show();
            {
                 send(0x05,0x5a);
            }
                break;
            case R.id.radio15:
                  // DisplayToast("灯光亮度 80%！");
                 Toast.makeText(Secondpage.this, "第五路灯光亮度为 80%！",
                                        Toast.LENGTH_SHORT).show();
            {
                 send(0x05,0x23);
            }
                break;
            default:
                  // DisplayToast("灯光亮度 100%！");
                 Toast.makeText(Secondpage.this, "第五路灯光亮度为 100%！",
                                        Toast.LENGTH_SHORT).show();
            {
                 send(0x05,0x00);
            }
                break;
            }

          }
    });

    radiogroup1=(RadioGroup)findViewById(R.id.radioGroup3);
    radio11=(RadioButton)findViewById(R.id.radio21);
    radio12=(RadioButton)findViewById(R.id.radio22);
    radio13=(RadioButton)findViewById(R.id.radio23);
    radio14=(RadioButton)findViewById(R.id.radio24);
```

```
radio15=(RadioButton)findViewById(R.id.radio25);
radio16=(RadioButton)findViewById(R.id.radio26);
radiogroup1.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        // TODO Auto-generated method stub

      switch (checkedId)
      {
      case R.id.radio21:
            /*DisplayToast("灯光亮度 0%！");*/
          Toast.makeText(Secondpage.this, "第六路灯光亮度为 0%！",
                    Toast.LENGTH_SHORT).show();
      {
          send(0x06,0xff);
      }
          break;
      case R.id.radio22:
           // DisplayToast("灯光亮度 20%！");
          Toast.makeText(Secondpage.this, "第六路灯光亮度为 20%！",
                         Toast.LENGTH_SHORT).show();
      {
          send(0x06,0xc8);
      }
          break;
      case R.id.radio23:
           //DisplayToast("灯光亮度 40%！");
          Toast.makeText(Secondpage.this, "第六路灯光亮度为 40%！",
                         Toast.LENGTH_SHORT).show();
      {
          send(0x06,0x91);
      }
          break;
      case R.id.radio24:
            //DisplayToast("灯光亮度 60%！");
          Toast.makeText(Secondpage.this, "第六路灯光亮度为 60%！",
                         Toast.LENGTH_SHORT).show();
      {
          send(0x06,0x5a);
      }
          break;
      case R.id.radio25:
           // DisplayToast("灯光亮度 80%！");
          Toast.makeText(Secondpage.this, "第六路灯光亮度为 80%！",
                         Toast.LENGTH_SHORT).show();
      {
          send(0x06,0x23);
      }
          break;
      default:
```

```
                    // DisplayToast("灯光亮度 100%！");
                    Toast.makeText(Secondpage.this, "第六路灯光亮度为 100%！",
                                    Toast.LENGTH_SHORT).show();
                {
                    send(0x06,0x00);
                }
                    break;
                }

            }
        });


        radiogroup2=(RadioGroup)findViewById(R.id.radioGroup4);
        radio21=(RadioButton)findViewById(R.id.radio31);
        radio22=(RadioButton)findViewById(R.id.radio32);
        radio23=(RadioButton)findViewById(R.id.radio33);
        radio24=(RadioButton)findViewById(R.id.radio34);
        radio25=(RadioButton)findViewById(R.id.radio35);
        radio26=(RadioButton)findViewById(R.id.radio36);
        radiogroup2.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                // TODO Auto-generated method stub

                switch (checkedId)
                {
                case R.id.radio31:
                     /*DisplayToast("灯光亮度 0%！");*/
                    Toast.makeText(Secondpage.this, "第七路灯光亮度为 0%！",
                              Toast.LENGTH_SHORT).show();
                {

                    send(0x07,0xff);
                }
                    break;
                case R.id.radio32:
                    // DisplayToast("灯光亮度 20%！");
                    Toast.makeText(Secondpage.this, "第七路灯光亮度为 20%！",
                                    Toast.LENGTH_SHORT).show();
                {
                    send(0x07,0xc8);
                }
                    break;
                case R.id.radio33:
                    //DisplayToast("灯光亮度 40%！");
                    Toast.makeText(Secondpage.this, "第七路灯光亮度为 40%！",
                                    Toast.LENGTH_SHORT).show();
                {
                    send(0x07,0x91);
```

```
            }
                break;
            case R.id.radio34:
                    //DisplayToast("灯光亮度 60%！");
                  Toast.makeText(Secondpage.this, "第七路灯光亮度为 60%！",
                                Toast.LENGTH_SHORT).show();
            {
                  send(0x07,0x5a);
            }
                break;
            case R.id.radio35:
                  // DisplayToast("灯光亮度 80%！");
                  Toast.makeText(Secondpage.this, "第七路灯光亮度为 80%！",
                                Toast.LENGTH_SHORT).show();
            {
                  send(0x07,0x23);
            }
                break;
            default:
                  // DisplayToast("灯光亮度 100%！");
                  Toast.makeText(Secondpage.this, "第七路灯光亮度为 100%！",
                                Toast.LENGTH_SHORT).show();
            {
                  send(0x07,0x00);
            }
                break;
            }

        }
    });     //第八路


    radiogroup3=(RadioGroup)findViewById(R.id.radioGroup5);
    radio31=(RadioButton)findViewById(R.id.radio41);
    radio32=(RadioButton)findViewById(R.id.radio42);
    radio33=(RadioButton)findViewById(R.id.radio43);
    radio34=(RadioButton)findViewById(R.id.radio44);
    radio35=(RadioButton)findViewById(R.id.radio45);
    radio36=(RadioButton)findViewById(R.id.radio46);
    radiogroup3.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

        @Override
        public void onCheckedChanged(RadioGroup group, int checkedId) {
            // TODO Auto-generated method stub

            switch (checkedId)
            {
            case R.id.radio41:
                    /*DisplayToast("灯光亮度 0%！");*/
                  Toast.makeText(Secondpage.this, "第八路灯光亮度为 0%！",
```

```
                        Toast.LENGTH_SHORT).show();
            {
                send(0x08,0xff);
            }
                break;
            case R.id.radio42:
                 // DisplayToast("灯光亮度 20%！");
                Toast.makeText(Secondpage.this, "第八路灯光亮度为 20%！",
                              Toast.LENGTH_SHORT).show();
            {
                send(0x08,0xc8);
            }
                break;
            case R.id.radio43:
                 //DisplayToast("灯光亮度 40%！");
                Toast.makeText(Secondpage.this, "第八路灯光亮度为 40%！",
                              Toast.LENGTH_SHORT).show();
            {
                send(0x08,0x91);
            }
                break;
            case R.id.radio44:
                  //DisplayToast("灯光亮度 60%！");
                Toast.makeText(Secondpage.this, "第八路灯光亮度为 60%！",
                              Toast.LENGTH_SHORT).show();
            {
                send(0x08,0x5a);
            }
                break;
            case R.id.radio45:
                 // DisplayToast("灯光亮度 80%！");
                Toast.makeText(Secondpage.this, "第八路灯光亮度为 80%！",
                              Toast.LENGTH_SHORT).show();
            {
                send(0x08,0x23);
            }
                break;
            default:
                 // DisplayToast("灯光亮度 100%！");
                Toast.makeText(Secondpage.this, "第八路灯光亮度为 100%！",
                              Toast.LENGTH_SHORT).show();
            {
                send(0x08,0x00);
            }
                break;
            }

        }
    });
```

```
        ibutton5_o=(ImageButton) findViewById(R.id.imageButton2);
        ibutton5_o.setOnClickListener(new ClickEventKey());
        ibutton5_c=(ImageButton) findViewById(R.id.imageButton3);
        ibutton5_c.setOnClickListener(new ClickEventKey());
        ibutton6_o=(ImageButton) findViewById(R.id.imageButton4);
        ibutton6_o.setOnClickListener(new ClickEventKey());
        ibutton6_c=(ImageButton) findViewById(R.id.imageButton5);
        ibutton6_c.setOnClickListener(new ClickEventKey());
        ibutton7_o=(ImageButton) findViewById(R.id.imageButton6);
        ibutton7_o.setOnClickListener(new ClickEventKey());
        ibutton7_c=(ImageButton) findViewById(R.id.imageButton7);
        ibutton7_c.setOnClickListener(new ClickEventKey());
        ibutton8_o=(ImageButton) findViewById(R.id.imageButton8);
        ibutton8_o.setOnClickListener(new ClickEventKey());
        ibutton8_c=(ImageButton) findViewById(R.id.imageButton9);
        ibutton8_c.setOnClickListener(new ClickEventKey());
 }

class ClickEventKey implements View.OnClickListener {
 @Override
 public void onClick(View v) {
//      if (v == ibutton1_o)
        switch (v.getId())

        {
        case R.id.imageButton2:
        {
            send(0x05,0x23);
        }

        break;
        case R.id.imageButton3:
            {
            send(0x05,0xff);
        }

        break;
        case R.id.imageButton4:
        {
            send(0x06,0x23);
        }

        break;
        case R.id.imageButton5:
        {
            send(0x06,0xff);
        }

        break;
            case R.id.imageButton6:
```

```
                {
                        send(0x07,0x23);
                }

                break;
                        case R.id.imageButton7:
                {
                        send(0x07,0xff);
                }

                break;
                        case R.id.imageButton8:
                {
                        send(0x08,0x23);
                }

                break;
                        case R.id.imageButton9:
                {
                        send(0x08,0xff);
                }

                break;
                }
                }
        }

private void    send(int Room,int Grade){

                try {

                           //String strpass="@@UP...";
                          byte[ ] byteone=new byte[8];//=strpass.getBytes("US-ASCII");
                                byteone[0]=(byte)0xf5;
                                byteone[1]=(byte)0x5f;
                                byteone[2]=(byte)0x00;
                                byteone[3]=(byte)Room;
                                byteone[4]=(byte)0x03;
                                byteone[5]=(byte)0x00;
                                byteone[6]=(byte)Grade;
                                byteone[7]=(byte)0x06;

                             //byte[ ] bytekai=strpass.getBytes("US-ASCII");
                      tmpOut = BluetoothMain.btSocket.getOutputStream();
                      tmpOut.write(byteone);}
                 catch (IOException e) {
                      Log.e("BluetoothReadService", "temp sockets not created", e);
                }
                }
}
```

到此为止，本实例的主要功能模块的实现过程介绍完毕。有关蓝牙方面的知识，将在本书后面的章节中进行讲解。本实例执行后的效果如图 11-1 所示。



图 11-1　执行效果

第 12 章　网络防火墙系统

本章的网络流量防火墙系统实例采用 Android 开源系统技术，利用 Java 语言和 Eclipse 开发工具对防火墙系统进行开发，同时给出详细的系统设计流程、部分界面图及主要功能效果流程图。在本章的内容中，还对开发过程中遇到的问题和解决方法进行详细的讨论。整个系统实例集允许上网、权限设置、系统帮助等功能于一体，在 Android 系统中能独立运行。在讲解具体编码之前，先简要介绍本项目的产生背景和项目意义，为后面的系统设计及编码工作做好准备。

12.1　系统需求分析

知识点讲解：光盘:视频\视频讲解\第 12 章\系统需求分析.avi

根据项目的目标，可分析出系统的基本需求，下面从软件设计的角度来介绍系统的功能，并且使用用例图来描述系统的功能模块，功能模块大致分成两部分来概括，分别是主界面和设置界面。主界面又可以细分为选择模式和勾选应用两部分，设置界面又可以细分为防火墙开关、日志开关、保存规则、退出、帮助和更多 6 个部分。整个系统的构成模块结构如图 12-1 所示。

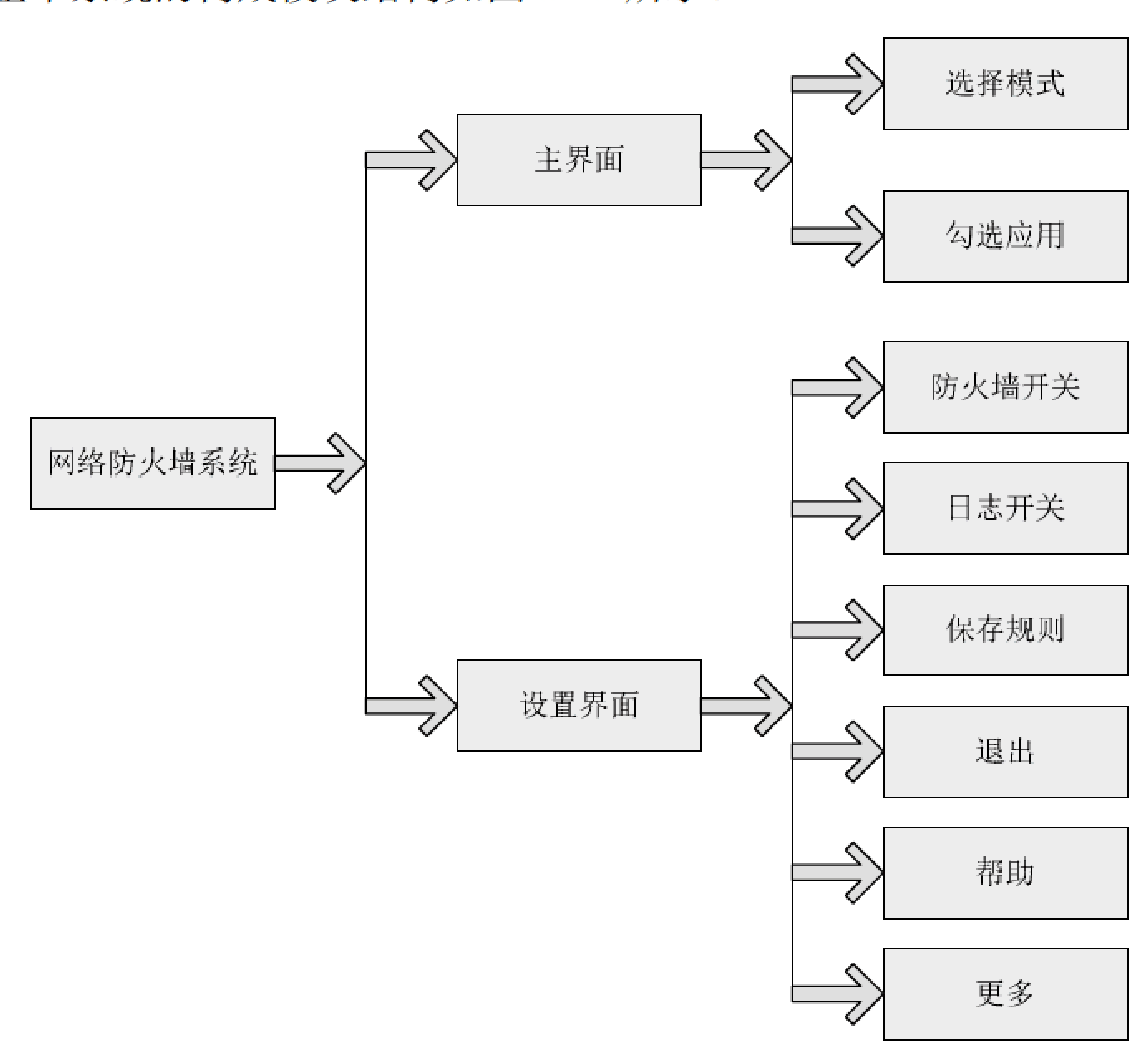


图 12-1　系统构成模块

（1）系统性能需求

根据 Android 手机系统要求无响应时间为 5 秒，有如下性能要求：

- ☑ 选择模式设置，程序响应时间最长不能超过 5 秒。
- ☑ 勾选应用设置，程序响应时间最长不能超过 5 秒。
- ☑ 防火墙开关设置，程序响应时间最长不能超过 5 秒。
- ☑ 日志开关设置，程序响应时间最长不能超过 5 秒。
- ☑ 保存规则设置，程序响应时间最长不能超过 5 秒。

（2）运行环境需求

- ☑ 操作系统：Android 手机基于 Linux 操作系统。
- ☑ 支持环境：Android 2.3 以上版本。
- ☑ 开发环境：Eclipse 3.5 ADT 0.95。

12.2　编写布局文件

知识点讲解：光盘:视频\视频讲解\第 12 章\编写布局文件.avi

（1）首先编写主界面文件 main.xml，系统执行之后首先显示主界面，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:layout_width="fill_parent"
    android:layout_height="fill_parent" xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:duplicateParentState="false">
    <View android:layout_width="fill_parent" android:layout_height="1sp"
        android:background="#FFFFFFFF" />
    <LinearLayout android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:padding="8sp">
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:id="@+id/label_mode"
            android:text="Mode: " android:textSize="20sp" android:clickable="true"></TextView>
    </LinearLayout>
    <View android:layout_width="fill_parent" android:layout_height="1sp"
        android:background="#FFFFFFFF" />
    <RelativeLayout android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:padding="3sp">
        <ImageView android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:id="@+id/img_wifi"
            android:src="@drawable/eth_wifi" android:clickable="false"
            android:layout_alignParentLeft="true" android:paddingLeft="3sp"
            android:paddingRight="10sp"></ImageView>
        <ImageView android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:id="@+id/img_3g"
            android:layout_toRightOf="@id/img_wifi" android:src="@drawable/eth_g"
            android:clickable="false"></ImageView>
        <ImageView android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:id="@+id/img_download"
            android:src="@drawable/download" android:layout_alignParentRight="true"
            android:paddingLeft="22sp" android:clickable="false"></ImageView>
        <ImageView android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:id="@+id/img_upload"
            android:layout_toLeftOf="@id/img_download" android:src="@drawable/upload"
```

```
                android:clickable="false"></ImageView>
        </RelativeLayout>
        <ListView android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:id="@+id/listview"></ListView>
</LinearLayout>
```

在上述代码中，将整个主界面划分为如下两个部分。

☑ 上部分：显示模式和网络类型，其中模式分为黑名单模式和白名单模式。

☑ 下部分：列表显示了某种模式下的所有网络服务，并且在每种服务前面显示一个复选框按钮，通过按钮可以设置某种服务启用还是禁用。

下部分的列表功能是通过文件 listitem.xml 实现的，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent" android:layout_height="fill_parent">
        <CheckBox android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:id="@+id/itemcheck_wifi"
            android:layout_alignParentLeft="true"></CheckBox>
        <CheckBox android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:id="@+id/itemcheck_3g"
            android:layout_toRightOf="@id/itemcheck_wifi"></CheckBox>
        <TextView android:layout_height="wrap_content" android:id="@+id/app_text"
            android:text="uid:packages" android:layout_width="match_parent"
            android:layout_toRightOf="@id/itemcheck_3g" android:layout_centerVertical="true"
            android:paddingRight="80sp"></TextView>
        <TextView android:layout_height="wrap_content" android:id="@+id/download"
            android:layout_width="wrap_content" android:layout_alignParentRight="true"
            android:layout_centerVertical="true" android:paddingLeft="15sp"></TextView>
        <TextView android:layout_height="wrap_content" android:id="@+id/upload"
            android:layout_width="wrap_content" android:layout_toLeftOf="@id/download"
            android:layout_centerVertical="true"></TextView>
</RelativeLayout>
```

系统主界面的效果如图 12-2 所示。

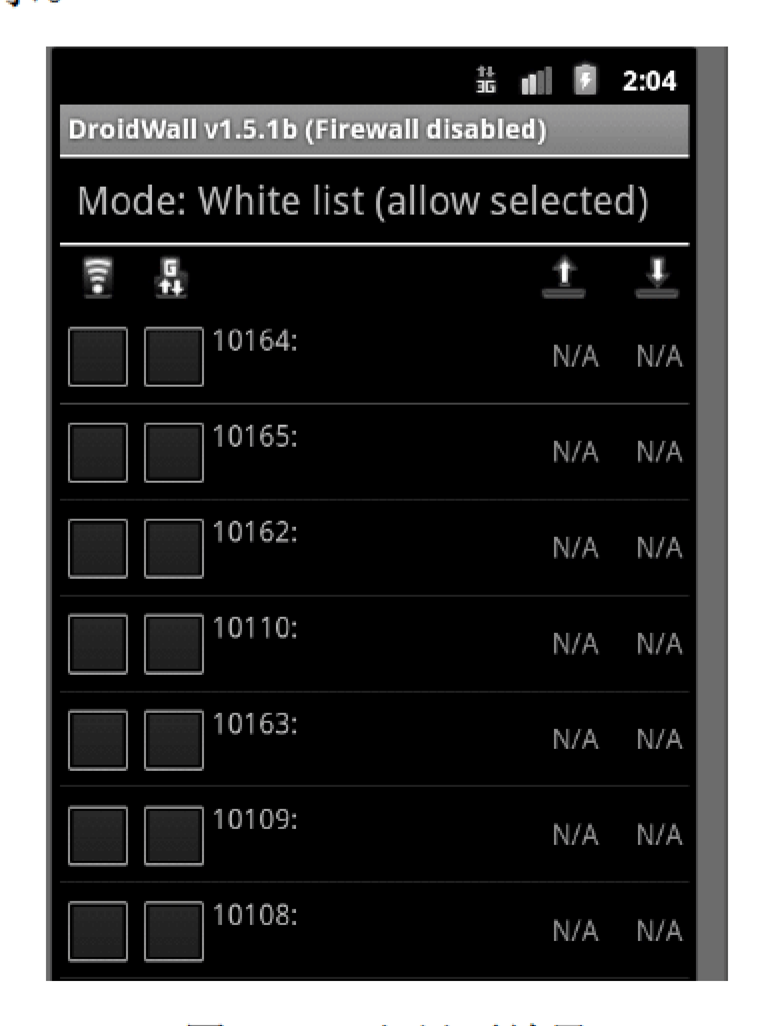
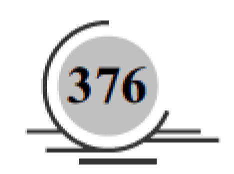


图 12-2　主界面效果



（2）编写帮助界面布局文件 help_dialog.xml，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
	android:layout_width="fill_parent" android:layout_height="wrap_content">
	<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
		android:layout_width="fill_parent" android:layout_height="fill_parent">
			<TextView android:layout_height="fill_parent"
				android:layout_width="fill_parent" android:text="@string/help_dialog_text"
				android:padding="6dip" />
	</ScrollView>
</FrameLayout>
```

系统帮助界面的效果如图 12-3 所示。

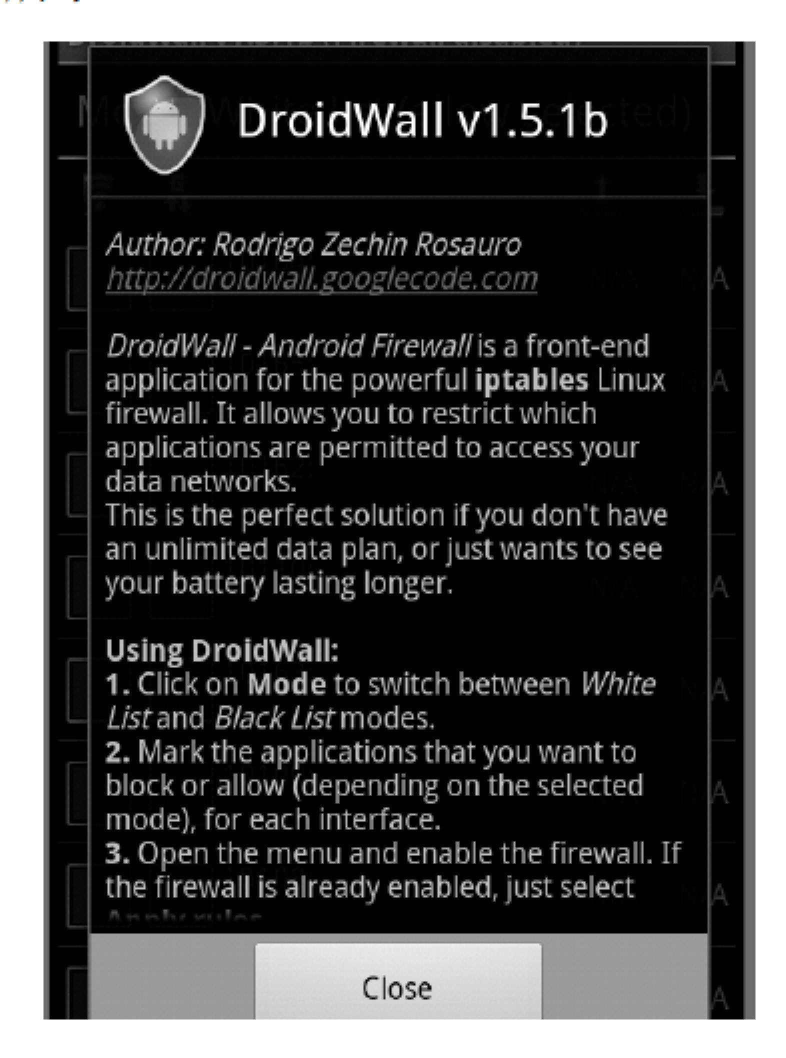


图 12-3　帮助界面效果

12.3　编写主程序文件

知识点讲解：光盘:视频\视频讲解\第 12 章\编写主程序文件.avi

布局文件编写完毕之后，还需要编写值文件 strings.xml，具体代码比较简单，请读者参考本书附带光盘中的代码即可，在此不再进行详细讲解。接下来讲解使用 Java 编写主程序文件的具体实现流程。

12.3.1　主 Activity 文件

首先编写文件 MainActivity.java，此文件是整个系统的核心，能够实现服务勾选处理和模式设置功能，选中后会禁止或开启某项网络服务。文件 MainActivity.java 的具体实现流程如下所示。

☑ 定义类 MainActivity 为项目启动后首先显示的 Activity，设置按下 Menu 后显示的选项，并设置需要的各个实例函数。部分代码如下所示。

```
/**
 * 主 Activity。当打开应用时，这是被显示的界面
 */
```

```
public class MainActivity extends Activity implements OnCheckedChangeListener,
        OnClickListener {
    // 按下 Menu 后显示的选项
    private static final int MENU_DISABLE = 0;
    private static final int MENU_TOGGLELOG = 1;
    private static final int MENU_APPLY = 2;
    private static final int MENU_EXIT = 3;
    private static final int MENU_HELP = 4;
    private static final int MENU_SHOWLOG = 5;
    private static final int MENU_SHOWRULES = 6;
    private static final int MENU_CLEARLOG = 7;
    private static final int MENU_SETPWD = 8;

    /**进展对话实例*/
    private ListView listview;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        checkPreferences();
        setContentView(R.layout.main);
        this.findViewById(R.id.label_mode).setOnClickListener(this);
        Api.assertBinaries(this, true);
    }

    @Override
    protected void onStart() {
        super.onStart();
        // Force re-loading the application list
        Log.d("DroidWall", "onStart() - Forcing APP list reload!");
        Api.applications = null;
    }
    @Override
    protected void onResume() {
        super.onResume();
        if (this.listview == null) {
            this.listview = (ListView) this.findViewById(R.id.listview);
        }
        refreshHeader();
        final String pwd = getSharedPreferences(Api.PREFS_NAME, 0).getString(
                Api.PREF_PASSWORD, "");
        if (pwd.length() == 0) {
            // No password lock
            showOrLoadApplications();
        } else {
            // Check the password
            requestPassword(pwd);
        }
    }
    @Override
    protected void onPause() {
        super.onPause();
```

```
            this.listview.setAdapter(null);
    }
```

☑ 定义函数 checkPreferences()检查被存放的选项正常，具体代码如下所示。

```
    /**
     * 检查被存放的选项正常
     */
    private void checkPreferences() {
            final SharedPreferences prefs = getSharedPreferences(Api.PREFS_NAME, 0);
            final Editor editor = prefs.edit();
            boolean changed = false;
            if (prefs.getString(Api.PREF_MODE, "").length() == 0) {
                    editor.putString(Api.PREF_MODE, Api.MODE_WHITELIST);
                    changed = true;
            }
            /*删除旧的选项名字*/
            if (prefs.contains("AllowedUids")) {
                    editor.remove("AllowedUids");
                    changed = true;
            }
            if (prefs.contains("Interfaces")) {
                    editor.remove("Interfaces");
                    changed = true;
            }
            if (changed)
                    editor.commit();
    }
```

☑ 定义函数 refreshHeader()刷新显示当前运行的和网络相关的程序，具体代码如下所示。

```
    /**
     * 刷新显示当前运行的和网络相关的程序
     */
    private void refreshHeader() {
            final SharedPreferences prefs = getSharedPreferences(Api.PREFS_NAME, 0);
            final String mode = prefs.getString(Api.PREF_MODE, Api.MODE_WHITELIST);
            final TextView labelmode = (TextView) this
                            .findViewById(R.id.label_mode);
            final Resources res = getResources();
            int resid = (mode.equals(Api.MODE_WHITELIST) ? R.string.mode_whitelist
                            : R.string.mode_blacklist);
            labelmode.setText(res.getString(R.string.mode_header,
                            res.getString(resid)));
            resid = (Api.isEnabled(this) ? R.string.title_enabled
                            : R.string.title_disabled);
            setTitle(res.getString(resid, Api.VERSION));
    }
```

☑ 定义函数 selectMode()显示对话框选择操作方式，供用户选择黑名单模式和白名单模式。具体代码如下所示。

```
    /**
     * 显示对话框选择操作方式，供用户选择黑名单模式和白名单模式
     */
    private void selectMode() {
```

```
            final Resources res = getResources();
            new AlertDialog.Builder(this)
                    .setItems(
                            new String[ ] { res.getString(R.string.mode_whitelist),
                                    res.getString(R.string.mode_blacklist) },
                            new DialogInterface.OnClickListener() {
                                public void onClick(DialogInterface dialog,
                                        int which) {
                                    final String mode = (which == 0 ? Api.MODE_WHITELIST
                                            : Api.MODE_BLACKLIST);
                                    final Editor editor = getSharedPreferences(
                                            Api.PREFS_NAME, 0).edit();
                                    editor.putString(Api.PREF_MODE, mode);
                                    editor.commit();
                                    refreshHeader();
                                }
                            }).setTitle("Select mode:").show();
    }
```

☑ 定义函数 setPassword()来设置一个系统密码，设置密码后，在进入主界面前会通过函数 requestPassword()来验证密码，只有密码正确才能进入。具体代码如下所示。

```
    /**
     * 设置一个新的密码
     */
    private void setPassword(String pwd) {
        final Resources res = getResources();
        final Editor editor = getSharedPreferences(Api.PREFS_NAME, 0).edit();
        editor.putString(Api.PREF_PASSWORD, pwd);
        String msg;
        if (editor.commit()) {
            if (pwd.length() > 0) {
                msg = res.getString(R.string.passdefined);
            } else {
                msg = res.getString(R.string.passremoved);
            }
        } else {
            msg = res.getString(R.string.passerror);
        }
        Toast.makeText(MainActivity.this, msg, Toast.LENGTH_SHORT).show();
    }

    /**
     * 如果设置了密码，显示主界面前先验证密码
     */
    private void requestPassword(final String pwd) {
        new PassDialog(this, false, new android.os.Handler.Callback() {
            public boolean handleMessage(Message msg) {
                if (msg.obj == null) {
                    MainActivity.this.finish();
                    android.os.Process.killProcess(android.os.Process.myPid());
                    return false;
```

```
                }
                if (!pwd.equals(msg.obj)) {
                    requestPassword(pwd);
                    return false;
                }
                //如果密码正确
                showOrLoadApplications();
                return false;
            }
        }).show();
    }
```

☑ 编写函数 toggleLogEnabled()实现防火墙禁用和日志禁用开关处理，具体代码如下所示。

```
    /**
     * 开关设置
     */
    private void toggleLogEnabled() {
        final SharedPreferences prefs = getSharedPreferences(Api.PREFS_NAME, 0);
        final boolean enabled = !prefs.getBoolean(Api.PREF_LOGENABLED, false);
        final Editor editor = prefs.edit();
        editor.putBoolean(Api.PREF_LOGENABLED, enabled);
        editor.commit();
        if (Api.isEnabled(this)) {
            Api.applySavedIptablesRules(this, true);
        }
        Toast.makeText(
                MainActivity.this,
                (enabled ? R.string.log_was_enabled : R.string.log_was_disabled),
                Toast.LENGTH_SHORT).show();
    }
```

☑ 编写函数 showOrLoadApplications()，如果在某模式下有应用，则显示应用。函数 showOrLoad Applications()的具体代码如下所示。

```
    /**
     * 如果某模式下有应用，则显示应用
     */
    private void showOrLoadApplications() {
        final Resources res = getResources();
        if (Api.applications == null) {
            final ProgressDialog progress = ProgressDialog.show(this,
                    res.getString(R.string.working),
                    res.getString(R.string.reading_apps), true);
            final Handler handler = new Handler() {
                public void handleMessage(Message msg) {
                    try {
                        progress.dismiss();
                    } catch (Exception ex) {
                    }
                    showApplications();
                }
            };
            new Thread() {
```

```
                public void run() {
                    Api.getApps(MainActivity.this);
                    handler.sendEmptyMessage(0);
                }
            }.start();
        } else {
            //存储应用，显示名单
            showApplications();
        }
    }
```

☑ 编写函数 showApplications()显示应用名单，具体代码如下所示。

```
    /**
     * 显示应用名单
     */
    private void showApplications() {
        final DroidApp[ ] apps = Api.getApps(this);
        // Sort applications - selected first, then alphabetically
        Arrays.sort(apps, new Comparator<DroidApp>() {
            @Override
            public int compare(DroidApp o1, DroidApp o2) {
                if ((o1.selected_wifi | o1.selected_3g) == (o2.selected_wifi | o2.selected_3g)) {
                    return String.CASE_INSENSITIVE_ORDER.compare(o1.names[0],
                            o2.names[0]);
                }
                if (o1.selected_wifi || o1.selected_3g)
                    return -1;
                return 1;
            }
        });
        final LayoutInflater inflater = getLayoutInflater();
        final ListAdapter adapter = new ArrayAdapter<DroidApp>(this,
                R.layout.listitem, R.id.app_text, apps) {
            @Override
            public View getView(int position, View convertView, ViewGroup parent) {
                ListEntry entry;
                if (convertView == null) {
                    // Inflate a new view
                    convertView = inflater.inflate(R.layout.listitem, parent,
                            false);
                    entry = new ListEntry();
                    entry.box_wifi = (CheckBox) convertView
                            .findViewById(R.id.itemcheck_wifi);
                    entry.box_3g = (CheckBox) convertView
                            .findViewById(R.id.itemcheck_3g);
                    entry.app_text = (TextView) convertView
                            .findViewById(R.id.app_text);
                    entry.upload = (TextView) convertView
                            .findViewById(R.id.upload);
                    entry.download = (TextView) convertView
                            .findViewById(R.id.download);
                    convertView.setTag(entry);
```

```
                    entry.box_wifi
                            .setOnCheckedChangeListener(MainActivity.this);
                    entry.box_3g.setOnCheckedChangeListener(MainActivity.this);
                } else {
                    //转换一个现有视图
                    entry = (ListEntry) convertView.getTag();
                }
                final DroidApp app = apps[position];
                entry.app_text.setText(app.toString());
                convertAndSetColor(TrafficStats.getUidTxBytes(app.uid), entry.upload);
                convertAndSetColor(TrafficStats.getUidRxBytes(app.uid), entry.download);
                final CheckBox box_wifi = entry.box_wifi;
                box_wifi.setTag(app);
                box_wifi.setChecked(app.selected_wifi);
                final CheckBox box_3g = entry.box_3g;
                box_3g.setTag(app);
                box_3g.setChecked(app.selected_3g);
                return convertView;
            }
```

☑ 编写函数 convertAndSetColor()，根据对某选项的设置显示内容，并设置显示内容的颜色。假如没有任何设置，则显示 N/A，如果已经设置了启用则显示已经用过的流量。函数 convertAndSetColor()的部分代码如下所示。

```
            private void convertAndSetColor(long num, TextView text) {
                String value = null;
                long temp = num;
                float floatnum = num;
                if (num == -1) {
                    value = "N/A ";
                    text.setText(value);
                    text.setTextColor(0xff919191);
                    return ;
                } else if ((temp = temp / 1024) < 1) {
                    value = num + "B";
                } else if ((floatnum = temp / 1024) < 1) {
                    value = temp + "KB";
                } else {
                    DecimalFormat format = new DecimalFormat("##0.0");
                    value = format.format(floatnum) + "MB";
                }
                text.setText(value);
                text.setTextColor(0xffff0300);
            }
        };
        this.listview.setAdapter(adapter);
    }
```

☑ 进入系统主界面后，如果按下 Menu 键则会弹出设置界面，在设置界面中可以选择对应的功能。在设置界面中的选择功能是通过如下 3 个函数实现的。

```
public boolean onCreateOptionsMenu(Menu menu) {
        menu.add(0, MENU_DISABLE, 0, R.string.fw_enabled).setIcon(
                android.R.drawable.button_onoff_indicator_on);
```

```
        menu.add(0, MENU_TOGGLELOG, 0, R.string.log_enabled).setIcon(
                android.R.drawable.button_onoff_indicator_on);
        menu.add(0, MENU_APPLY, 0, R.string.applyrules).setIcon(
                R.drawable.apply);
        menu.add(0, MENU_EXIT, 0, R.string.exit).setIcon(
                android.R.drawable.ic_menu_close_clear_cancel);
        menu.add(0, MENU_HELP, 0, R.string.help).setIcon(
                android.R.drawable.ic_menu_help);
        menu.add(0, MENU_SHOWLOG, 0, R.string.show_log)
                .setIcon(R.drawable.show);
        menu.add(0, MENU_SHOWRULES, 0, R.string.showrules).setIcon(
                R.drawable.show);
        menu.add(0, MENU_CLEARLOG, 0, R.string.clear_log).setIcon(
                android.R.drawable.ic_menu_close_clear_cancel);
        menu.add(0, MENU_SETPWD, 0, R.string.setpwd).setIcon(
                android.R.drawable.ic_lock_lock);
        return true;
    }

    @Override
    public boolean onPrepareOptionsMenu(Menu menu) {
        final MenuItem item_onoff = menu.getItem(MENU_DISABLE);
        final MenuItem item_apply = menu.getItem(MENU_APPLY);
        final boolean enabled = Api.isEnabled(this);
        if (enabled) {
            item_onoff.setIcon(android.R.drawable.button_onoff_indicator_on);
            item_onoff.setTitle(R.string.fw_enabled);
            item_apply.setTitle(R.string.applyrules);
        } else {
            item_onoff.setIcon(android.R.drawable.button_onoff_indicator_off);
            item_onoff.setTitle(R.string.fw_disabled);
            item_apply.setTitle(R.string.saverules);
        }
        final MenuItem item_log = menu.getItem(MENU_TOGGLELOG);
        final boolean logenabled = getSharedPreferences(Api.PREFS_NAME, 0)
                .getBoolean(Api.PREF_LOGENABLED, false);
        if (logenabled) {
            item_log.setIcon(android.R.drawable.button_onoff_indicator_on);
            item_log.setTitle(R.string.log_enabled);
        } else {
            item_log.setIcon(android.R.drawable.button_onoff_indicator_off);
            item_log.setTitle(R.string.log_disabled);
        }
        return super.onPrepareOptionsMenu(menu);
    }

    @Override
    public boolean onMenuItemSelected(int featureId, MenuItem item) {
        switch (item.getItemId()) {
        case MENU_DISABLE:
            disableOrEnable();
```

```
                return true;
            case MENU_TOGGLELOG:
                toggleLogEnabled();
                return true;
            case MENU_APPLY:
                applyOrSaveRules();
                return true;
            case MENU_EXIT:
                finish();
                System.exit(0);
                return true;
            case MENU_HELP:
                new HelpDialog(this).show();
                return true;
            case MENU_SETPWD:
                setPassword();
                return true;
            case MENU_SHOWLOG:
                showLog();
                return true;
            case MENU_SHOWRULES:
                showRules();
                return true;
            case MENU_CLEARLOG:
                clearLog();
                return true;
            }
            return false;
        }
```

☑ 编写函数 disableOrEnable()设置开启或关闭防火墙，具体代码如下所示。

```
        private void disableOrEnable() {
            final boolean enabled = !Api.isEnabled(this);
            Log.d("DroidWall", "Changing enabled status to: " + enabled);
            Api.setEnabled(this, enabled);
            if (enabled) {
                applyOrSaveRules();
            } else {
                purgeRules();
            }
            refreshHeader();
        }
```

☑ 编写函数 setPassword()来到设置密码界面，具体代码如下所示。

```
        private void setPassword() {
            new PassDialog(this, true, new android.os.Handler.Callback() {
                public boolean handleMessage(Message msg) {
                    if (msg.obj != null) {
                        setPassword((String) msg.obj);
                    }
                    return false;
                }
```

```
            }).show();
    }
```

☑ 选择 Save rules（保存规则）后执行函数 showRules()，具体代码如下所示。

```
    private void showRules() {
        final Resources res = getResources();
        final ProgressDialog progress = ProgressDialog.show(this,
                res.getString(R.string.working),
                res.getString(R.string.please_wait), true);
        final Handler handler = new Handler() {
            public void handleMessage(Message msg) {
                try {
                    progress.dismiss();
                } catch (Exception ex) {
                }
                if (!Api.hasRootAccess(MainActivity.this, true))
                    return;
                Api.showIptablesRules(MainActivity.this);
            }
        };
        handler.sendEmptyMessageDelayed(0, 100);
    }
```

☑ 编写函数 showLog()显示日志信息界面，具体代码如下所示。

```
    private void showLog() {
        final Resources res = getResources();
        final ProgressDialog progress = ProgressDialog.show(this,
                res.getString(R.string.working),
                res.getString(R.string.please_wait), true);
        final Handler handler = new Handler() {
            public void handleMessage(Message msg) {
                try {
                    progress.dismiss();
                } catch (Exception ex) {
                }
                Api.showLog(MainActivity.this);
            }
        };
        handler.sendEmptyMessageDelayed(0, 100);
    }
```

☑ 编写函数 clearLog()清除系统内的日志记录信息，具体代码如下所示。

```
    private void clearLog() {
        final Resources res = getResources();
        final ProgressDialog progress = ProgressDialog.show(this,
                res.getString(R.string.working),
                res.getString(R.string.please_wait), true);
        final Handler handler = new Handler() {
            public void handleMessage(Message msg) {
                try {
                    progress.dismiss();
                } catch (Exception ex) {
                }
```

```
                if (!Api.hasRootAccess(MainActivity.this, true))
                    return;
                if (Api.clearLog(MainActivity.this)) {
                    Toast.makeText(MainActivity.this, R.string.log_cleared,
                            Toast.LENGTH_SHORT).show();
                }
            }
        };
        handler.sendEmptyMessageDelayed(0, 100);
    }
```

☑ 编写函数 applyOrSaveRules()，当申请或保存规则后将规则运用到本系统。具体代码如下所示。

```
    private void applyOrSaveRules() {
        final Resources res = getResources();
        final boolean enabled = Api.isEnabled(this);
        final ProgressDialog progress = ProgressDialog.show(this, res
                .getString(R.string.working), res
                .getString(enabled ? R.string.applying_rules
                        : R.string.saving_rules), true);
        final Handler handler = new Handler() {
            public void handleMessage(Message msg) {
                try {
                    progress.dismiss();
                } catch (Exception ex) {
                }
                if (enabled) {
                    Log.d("DroidWall", "Applying rules.");
                    if (Api.hasRootAccess(MainActivity.this, true)
                            && Api.applyIptablesRules(MainActivity.this, true)) {
                        Toast.makeText(MainActivity.this,
                                R.string.rules_applied, Toast.LENGTH_SHORT)
                                .show();
                    } else {
                        Log.d("DroidWall", "Failed - Disabling firewall.");
                        Api.setEnabled(MainActivity.this, false);
                    }
                } else {
                    Log.d("DroidWall", "Saving rules.");
                    Api.saveRules(MainActivity.this);
                    Toast.makeText(MainActivity.this, R.string.rules_saved,
                            Toast.LENGTH_SHORT).show();
                }
            }
        };
        handler.sendEmptyMessageDelayed(0, 100);
    }
```

☑ 编写函数 purgeRules()来清除一个规则，具体代码如下所示。

```
    private void purgeRules() {
        final Resources res = getResources();
        final ProgressDialog progress = ProgressDialog.show(this,
                res.getString(R.string.working),
```

```
                    res.getString(R.string.deleting_rules), true);
        final Handler handler = new Handler() {
            public void handleMessage(Message msg) {
                try {
                    progress.dismiss();
                } catch (Exception ex) {
                }
                if (!Api.hasRootAccess(MainActivity.this, true))
                    return;
                if (Api.purgeIptables(MainActivity.this, true)) {
                    Toast.makeText(MainActivity.this, R.string.rules_deleted,
                            Toast.LENGTH_SHORT).show();
                }
            }
        };
        handler.sendEmptyMessageDelayed(0, 100);
    }
```

☑ 编写函数 onCheckedChanged()检查 WI-FI 选项和 3G 选项是否发生变化。具体代码如下所示。

```
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        final DroidApp app = (DroidApp) buttonView.getTag();
        if (app != null) {
            switch (buttonView.getId()) {
            case R.id.itemcheck_wifi:
                app.selected_wifi = isChecked;
                break;
            case R.id.itemcheck_3g:
                app.selected_3g = isChecked;
                break;
            }
        }
    }
```

到此为止，主界面程序介绍完毕，按下 Menu 键后会弹出设置界面，如图 12-4 所示。

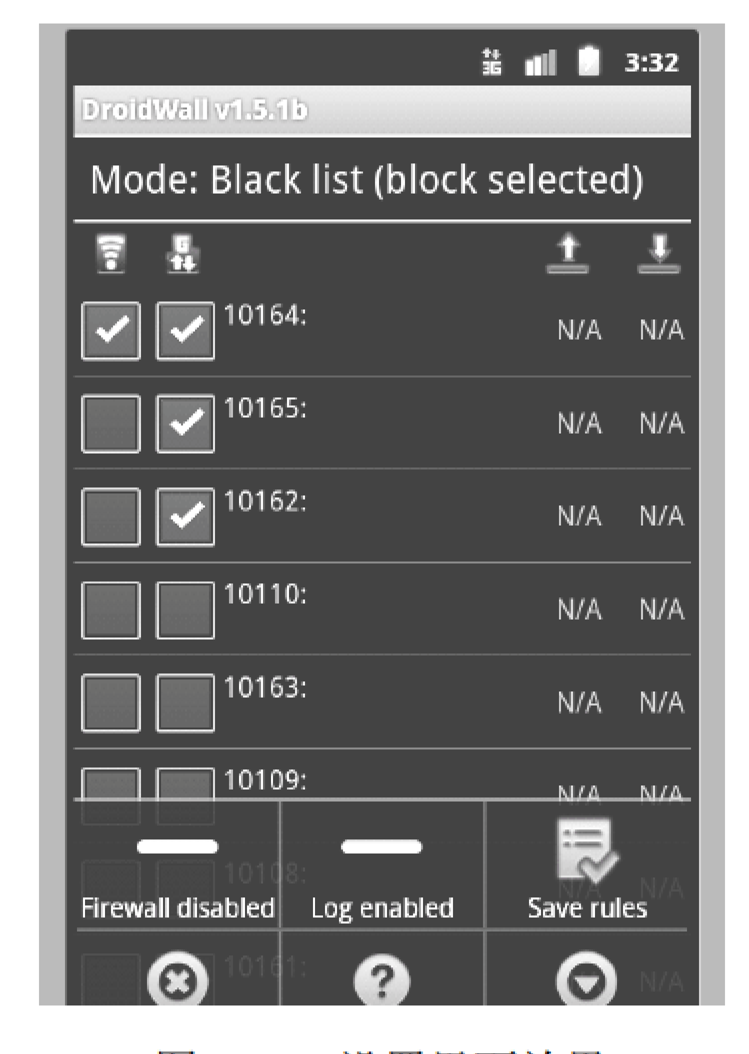


图 12-4 设置界面效果

12.3.2 帮助 Activity 文件

编写文件 HelpDialog.java，单击设置面板中的 ? 按钮后将会弹出帮助界面。文件 HelpDialog.java 的具体代码如下所示。

```
import android.app.AlertDialog;
import android.content.Context;
import android.view.View;
public class HelpDialog extends AlertDialog {
	protected HelpDialog(Context context) {
		super(context);
		final View view = getLayoutInflater().inflate(R.layout.help_dialog, null);
		setButton(context.getText(R.string.close), (OnClickListener)null);
		setIcon(R.drawable.icon);
		setTitle("DroidWall v" + Api.VERSION);
		setView(view);
	}
}
```

12.3.3 公共库函数文件

编写文件 Api.java，在此文件中定义了项目中需要的公共库函数。为了便于项目的开发，专门用此文件保存了系统中经常用到的函数。文件 Api.java 的具体代码如下所示。

☑ 编写函数 scriptHeader()创建一个通用的 Script 程序头，此程序可供二进制数据使用。具体代码如下所示。

```
	private static String scriptHeader(Context ctx) {
		final String dir = ctx.getDir("bin", 0).getAbsolutePath();
		final String myiptables = dir + "/iptables_armv5";
		return "" + "IPTABLES=iptables\n" + "BUSYBOX=busybox\n" + "GREP=grep\n"
				+ "ECHO=echo\n" + "# Try to find busybox\n" + "if "
				+ dir
				+ "/busybox_g1 --help >/dev/null 2>/dev/null ; then\n"
				+ " BUSYBOX="
				+ dir
				+ "/busybox_g1\n"
				+ " GREP=\"$BUSYBOX grep\"\n"
				+ " ECHO=\"$BUSYBOX echo\"\n"
				+ "elif busybox --help >/dev/null 2>/dev/null ; then\n"
				+ " BUSYBOX=busybox\n"
				+ "elif /system/xbin/busybox --help >/dev/null 2>/dev/null ; then\n"
				+ " BUSYBOX=/system/xbin/busybox\n"
				+ "elif /system/bin/busybox --help >/dev/null 2>/dev/null ; then\n"
				+ " BUSYBOX=/system/bin/busybox\n"
				+ "fi\n"
				+ "# Try to find grep\n"
				+ "if ! $ECHO 1 | $GREP -q 1 >/dev/null 2>/dev/null ; then\n"
				+ " if $ECHO 1 | $BUSYBOX grep -q 1 >/dev/null 2>/dev/null ; then\n"
				+ " GREP=\"$BUSYBOX grep\"\n"
				+ " fi\n"
```

```
                    + " # Grep is absolutely required\n"
                    + " if ! $ECHO 1 | $GREP -q 1 >/dev/null 2>/dev/null ; then\n"
                    + " $ECHO The grep command is required. DroidWall will not work.\n"
                    + " exit 1\n"
                    + "fi\n"
                    + "fi\n"
                    + "# Try to find iptables\n"
                    + "if "
                    + myiptables
                    + " --version >/dev/null 2>/dev/null ; then\n"
                    + " IPTABLES="
                    + myiptables + "\n" + "fi\n" + "";
}
```

☑ 编写函数 copyRawFile()，根据其 ID 给定的资源文件位置，复制一个未加工的资源文件。具体代码如下所示。

```
private static void copyRawFile(Context ctx, int resid, File file,
          String mode) throws IOException, InterruptedException {
     final String abspath = file.getAbsolutePath();
     //在 iptables 中写入二进制数据
     final FileOutputStream out = new FileOutputStream(file);
     final InputStream is = ctx.getResources().openRawResource(resid);
     byte buf[ ] = new byte[1024];
     int len;
     while ((len = is.read(buf)) > 0) {
          out.write(buf, 0, len);
     }
     out.close();
     is.close();
     //允许改变
     Runtime.getRuntime().exec("chmod " + mode + " " + abspath).waitFor();
}
```

☑ 编写函数 applyIptablesRulesImpl()，功能是清除并且重新加写所有规则，此功能是在内部实施的。函数 applyIptablesRulesImpl()的具体代码如下所示。

```
private static boolean applyIptablesRulesImpl(Context ctx,
          List<Integer> uidsWifi, List<Integer> uids3g, boolean showErrors) {
     if (ctx == null) {
          return false;
     }
     assertBinaries(ctx, showErrors);
     final String ITFS_WIFI[ ] = { "tiwlan+", "wlan+", "eth+" };
     final String ITFS_3G[ ] = { "rmnet+", "pdp+", "ppp+", "uwbr+", "wimax+","vsnet+" };
     final SharedPreferences prefs = ctx.getSharedPreferences(PREFS_NAME, 0);
     final boolean whitelist = prefs.getString(PREF_MODE, MODE_WHITELIST)
               .equals(MODE_WHITELIST);
     final boolean blacklist = !whitelist;
     final boolean logenabled = ctx.getSharedPreferences(PREFS_NAME, 0)
               .getBoolean(PREF_LOGENABLED, false);
     final StringBuilder script = new StringBuilder();
     try {
          int code;
```

```
script.append(scriptHeader(ctx));
script.append(""
        + "$IPTABLES --version || exit 1\n"
        + "# Create the droidwall chains if necessary\n"
        + "$IPTABLES -L droidwall >/dev/null 2>/dev/null || $IPTABLES --new droidwall || exit 2\n"
        + "$IPTABLES -L droidwall-3g >/dev/null 2>/dev/null || $IPTABLES --new
        droidwall-3g || exit 3\n"
        + "$IPTABLES -L droidwall-wifi >/dev/null 2>/dev/null || $IPTABLES --new
        droidwall-wifi || exit 4\n"
        + "$IPTABLES -L droidwall-reject >/dev/null 2>/dev/null || $IPTABLES --new
        droidwall-reject || exit 5\n"
        + "# Add droidwall chain to OUTPUT chain if necessary\n"
        + "$IPTABLES -L OUTPUT | $GREP -q droidwall || $IPTABLES -A OUTPUT
        -j droidwall || exit 6\n"
        + "# Flush existing rules\n"
        + "$IPTABLES -F droidwall || exit 7\n"
        + "$IPTABLES -F droidwall-3g || exit 8\n"
        + "$IPTABLES -F droidwall-wifi || exit 9\n"
        + "$IPTABLES -F droidwall-reject || exit 10\n" + "");
//检查是否能设置
if (logenabled) {
    script.append(""
            + "# Create the log and reject rules (ignore errors on the LOG target just in case
              it is not available)\n"
            + "$IPTABLES -A droidwall-reject -j LOG --log-prefix \"[DROIDWALL] \"
            --log-uid\n"
            + "$IPTABLES -A droidwall-reject -j REJECT || exit 11\n"
            + "");
} else {
    script.append(""
            + "# Create the reject rule (log disabled)\n"
            + "$IPTABLES -A droidwall-reject -j REJECT || exit 11\n"
            + "");
}
if (whitelist && logenabled) {
    script.append("# Allow DNS lookups on white-list for a better logging (ignore errors)\n");
    script.append("$IPTABLES -A droidwall -p udp --dport 53 -j RETURN\n");
}
script.append("# Main rules (per interface)\n");
for (final String itf : ITFS_3G) {
    script.append("$IPTABLES -A droidwall -o ").append(itf)
            .append(" -j droidwall-3g || exit\n");
}
for (final String itf : ITFS_WIFI) {
    script.append("$IPTABLES -A droidwall -o ").append(itf)
            .append(" -j droidwall-wifi || exit\n");
}
script.append("# Filtering rules\n");
final String targetRule = (whitelist ? "RETURN"
        : "droidwall-reject");
final boolean any_3g = uids3g.indexOf(SPECIAL_UID_ANY) >= 0;
```

```
final boolean any_wifi = uidsWifi.indexOf(SPECIAL_UID_ANY) >= 0;
if (whitelist && !any_wifi) {
	/*当设置开启 WI-FI 时需要保证用户允许 DHCP 和 WI-FI 功能*/
	int uid = android.os.Process.getUidForName("dhcp");
	if (uid != -1) {
		script.append("# dhcp user\n");
		script.append(
				"$IPTABLES -A droidwall-wifi -m owner --uid-owner ")
				.append(uid).append(" -j RETURN || exit\n");
	}
	uid = android.os.Process.getUidForName("wifi");
	if (uid != -1) {
		script.append("# wifi user\n");
		script.append(
				"$IPTABLES -A droidwall-wifi -m owner --uid-owner ")
				.append(uid).append(" -j RETURN || exit\n");
	}
}
if (any_3g) {
	if (blacklist) {
		/* block any application on this interface */
		script.append("$IPTABLES -A droidwall-3g -j ")
				.append(targetRule).append(" || exit\n");
	}
} else {
	/*释放或阻拦在这个接口的各自的应用*/
	for (final Integer uid : uids3g) {
		if (uid >= 0)
			script.append(
					"$IPTABLES -A droidwall-3g -m owner --uid-owner ")
					.append(uid).append(" -j ").append(targetRule)
					.append(" || exit\n");
	}
}
if (any_wifi) {
	if (blacklist) {
		/*阻拦在这个接口的所有应用*/
		script.append("$IPTABLES -A droidwall-wifi -j ")
				.append(targetRule).append(" || exit\n");
	}
} else {
	/*释放或阻拦在这个接口的各自的应用*/
	for (final Integer uid : uidsWifi) {
		if (uid >= 0)
			script.append(
					"$IPTABLES -A droidwall-wifi -m owner --uid-owner ")
					.append(uid).append(" -j ").append(targetRule)
					.append(" || exit\n");
	}
}
if (whitelist) {
```

```
				if (!any_3g) {
					if (uids3g.indexOf(SPECIAL_UID_KERNEL) >= 0) {
						script.append("# hack to allow kernel packets on white-list\n");
						script.append("$IPTABLES -A droidwall-3g -m owner --uid-owner 0:999999999
						-j droidwall-reject || exit\n");
					} else {
						script.append("$IPTABLES -A droidwall-3g -j droidwall-reject || exit\n");
					}
				}
				if (!any_wifi) {
					if (uidsWifi.indexOf(SPECIAL_UID_KERNEL) >= 0) {
						script.append("# hack to allow kernel packets on white-list\n");
						script.append("$IPTABLES -A droidwall-wifi -m owner --uid-owner 0:999999999
						-j droidwall-reject || exit\n");
					} else {
						script.append("$IPTABLES -A droidwall-wifi -j droidwall-reject || exit\n");
					}
				}
			} else {
				if (uids3g.indexOf(SPECIAL_UID_KERNEL) >= 0) {
					script.append("# hack to BLOCK kernel packets on black-list\n");
					script.append("$IPTABLES -A droidwall-3g -m owner --uid-owner 0:999999999
					-j RETURN || exit\n");
					script.append("$IPTABLES -A droidwall-3g -j droidwall-reject || exit\n");
				}
				if (uidsWifi.indexOf(SPECIAL_UID_KERNEL) >= 0) {
					script.append("# hack to BLOCK kernel packets on black-list\n");
					script.append("$IPTABLES -A droidwall-wifi -m owner --uid-owner 0:999999999
					-j RETURN || exit\n");
					script.append("$IPTABLES -A droidwall-wifi -j droidwall-reject || exit\n");
				}
			}
			final StringBuilder res = new StringBuilder();
			code = runScriptAsRoot(ctx, script.toString(), res);
			if (showErrors && code != 0) {
				String msg = res.toString();
				Log.e("DroidWall", msg);
				//去除多余的帮助信息
				if (msg.indexOf("\nTry `iptables -h' or 'iptables --help' for more information.") != -1) {
					msg = msg
							.replace(
									"\nTry `iptables -h' or 'iptables --help' for more information.",
									"");
				}
				alert(ctx, "Error applying iptables rules. Exit code: " + code
						+ "\n\n" + msg.trim());
			} else {
				return true;
			}
		} catch (Exception e) {
			if (showErrors)
```

```
                alert(ctx, "error refreshing iptables: " + e);
        }
        return false;
    }
```

☑ 编写函数 applySavedIptablesRules()，功能是清除并且重新加写所有规则，此规则不是内存中保存的。因为不需要读安装应用程序，所以比使用函数 applyIptablesRulesImpl()快。函数 applySavedIptablesRules()的具体代码如下所示。

```
    public static boolean applySavedIptablesRules(Context ctx,
                boolean showErrors) {
        if (ctx == null) {
            return false;
        }
        final SharedPreferences prefs = ctx.getSharedPreferences(PREFS_NAME, 0);
        final String savedUids_wifi = prefs.getString(PREF_WIFI_UIDS, "");
        final String savedUids_3g = prefs.getString(PREF_3G_UIDS, "");
        final List<Integer> uids_wifi = new LinkedList<Integer>();
        if (savedUids_wifi.length() > 0) {
            //检查哪些应用使用 WI-FI
            final StringTokenizer tok = new StringTokenizer(savedUids_wifi, "|");
            while (tok.hasMoreTokens()) {
                final String uid = tok.nextToken();
                if (!uid.equals("")) {
                    try {
                        uids_wifi.add(Integer.parseInt(uid));
                    } catch (Exception ex) {
                    }
                }
            }
        }
        final List<Integer> uids_3g = new LinkedList<Integer>();
        if (savedUids_3g.length() > 0) {
            //检查哪些应用允许 2G/3G 服务
            final StringTokenizer tok = new StringTokenizer(savedUids_3g, "|");
            while (tok.hasMoreTokens()) {
                final String uid = tok.nextToken();
                if (!uid.equals("")) {
                    try {
                        uids_3g.add(Integer.parseInt(uid));
                    } catch (Exception ex) {
                    }
                }
            }
        }
        return applyIptablesRulesImpl(ctx, uids_wifi, uids_3g, showErrors);
    }
```

☑ 编写函数 saveRules()，根据设置的选择项保存当前的规则，具体代码如下所示。

```
    public static void saveRules(Context ctx) {
        final SharedPreferences prefs = ctx.getSharedPreferences(PREFS_NAME, 0);
        final DroidApp[ ] apps = getApps(ctx);
        //建立被隔离的名单列表
```

```
        final StringBuilder newuids_wifi = new StringBuilder();
        final StringBuilder newuids_3g = new StringBuilder();
        for (int i = 0; i < apps.length; i++) {
            if (apps[i].selected_wifi) {
                if (newuids_wifi.length() != 0)
                    newuids_wifi.append('|');
                newuids_wifi.append(apps[i].uid);
            }
            if (apps[i].selected_3g) {
                if (newuids_3g.length() != 0)
                    newuids_3g.append('|');
                newuids_3g.append(apps[i].uid);
            }
        }
        //除 UIDs 新的名单之外
        final Editor edit = prefs.edit();
        edit.putString(PREF_WIFI_UIDS, newuids_wifi.toString());
        edit.putString(PREF_3G_UIDS, newuids_3g.toString());
        edit.commit();
    }
```

☑ 编写函数 purgeIptables()清除所有的过滤规则，具体代码如下所示。

```
    public static boolean purgeIptables(Context ctx, boolean showErrors) {
        StringBuilder res = new StringBuilder();
        try {
            assertBinaries(ctx, showErrors);
            int code = runScriptAsRoot(ctx, scriptHeader(ctx)
                    + "$IPTABLES -F droidwall\n"
                    + "$IPTABLES -F droidwall-reject\n"
                    + "$IPTABLES -F droidwall-3g\n"
                    + "$IPTABLES -F droidwall-wifi\n", res);
            if (code == -1) {
                if (showErrors)
                    alert(ctx, "error purging iptables. exit code: " + code
                            + "\n" + res);
                return false;
            }
            return true;
        } catch (Exception e) {
            if (showErrors)
                alert(ctx, "error purging iptables: " + e);
            return false;
        }
    }
```

☑ 编写函数 clearLog()清除系统中的日志记录信息，具体代码如下所示。

```
    public static boolean clearLog(Context ctx) {
        try {
            final StringBuilder res = new StringBuilder();
            int code = runScriptAsRoot(ctx, "dmesg -c >/dev/null || exit\n",
                    res);
            if (code != 0) {
```

```
                alert(ctx, res);
                return false;
            }
            return true;
        } catch (Exception e) {
            alert(ctx, "error: " + e);
        }
        return false;
    }
```

☑ 编写函数 showLog()显示系统中的日志记录信息，具体代码如下所示。

```
    public static void showLog(Context ctx) {
        try {
            StringBuilder res = new StringBuilder();
            int code = runScriptAsRoot(ctx, scriptHeader(ctx)
                    + "dmesg | $GREP DROIDWALL\n", res);
            if (code != 0) {
                if (res.length() == 0) {
                    res.append("Log is empty");
                }
                alert(ctx, res);
                return;
            }
            final BufferedReader r = new BufferedReader(new StringReader(
                    res.toString()));
            final Integer unknownUID = -99;
            res = new StringBuilder();
            String line;
            int start, end;
            Integer appid;
            final HashMap<Integer, LogInfo> map = new HashMap<Integer, LogInfo>();
            LogInfo loginfo = null;
            while ((line = r.readLine()) != null) {
                if (line.indexOf("[DROIDWALL]") == -1)
                    continue;
                appid = unknownUID;
                if (((start = line.indexOf("UID=")) != -1)
                        && ((end = line.indexOf(" ", start)) != -1)) {
                    appid = Integer.parseInt(line.substring(start + 4, end));
                }
                loginfo = map.get(appid);
                if (loginfo == null) {
                    loginfo = new LogInfo();
                    map.put(appid, loginfo);
                }
                loginfo.totalBlocked += 1;
                if (((start = line.indexOf("DST=")) != -1)
                        && ((end = line.indexOf(" ", start)) != -1)) {
                    String dst = line.substring(start + 4, end);
                    if (loginfo.dstBlocked.containsKey(dst)) {
                        loginfo.dstBlocked.put(dst,
                                loginfo.dstBlocked.get(dst) + 1);
```

```
                    } else {
                        loginfo.dstBlocked.put(dst, 1);
                    }
                }
            }
            final DroidApp[ ] apps = getApps(ctx);
            for (Integer id : map.keySet()) {
                res.append("App ID ");
                if (id != unknownUID) {
                    res.append(id);
                    for (DroidApp app : apps) {
                        if (app.uid == id) {
                            res.append(" (").append(app.names[0]);
                            if (app.names.length > 1) {
                                res.append(", ...)");
                            } else {
                                res.append(")");
                            }
                            break;
                        }
                    }
                } else {
                    res.append("(kernel)");
                }
                loginfo = map.get(id);
                res.append(" - Blocked ").append(loginfo.totalBlocked)
                        .append(" packets");
                if (loginfo.dstBlocked.size() > 0) {
                    res.append(" (");
                    boolean first = true;
                    for (String dst : loginfo.dstBlocked.keySet()) {
                        if (!first) {
                            res.append(", ");
                        }
                        res.append(loginfo.dstBlocked.get(dst))
                                .append(" packets for ").append(dst);
                        first = false;
                    }
                    res.append(")");
                }
                res.append("\n\n");
            }
            if (res.length() == 0) {
                res.append("Log is empty");
            }
            alert(ctx, res);
        } catch (Exception e) {
            alert(ctx, "error: " + e);
        }
    }
```

☑　编写函数 hasRootAccess()，检查是否具备进入根目录的权限，具体代码如下所示。

```
public static boolean hasRootAccess(Context ctx, boolean showErrors) {
        if (hasroot)
                return true;
        final StringBuilder res = new StringBuilder();
        try {
                // Run an empty script just to check root access
                if (runScriptAsRoot(ctx, "exit 0", res) == 0) {
                        hasroot = true;
                        return true;
                }
        } catch (Exception e) {
        }
        if (showErrors) {
                alert(ctx,
                                "Could not acquire root access.\n"
                                                + "You need a rooted phone to run DroidWall.\n\n"
                                                + "If this phone is already rooted, please make sure DroidWall has enough
                                                permissions to execute the \"su\" command.\n"
                                                + "Error message: " + res.toString());
        }
        return false;
}
```

☑ 编写函数 runScript()，执行前面编写的 Script 脚本头程序，此函数比较具有代表意义，能够在 Android 中调用并执行 Script 程序。函数 runScript()的具体代码如下所示。

```
public static int runScript(Context ctx, String script, StringBuilder res,
                long timeout, boolean asroot) {
        final File file = new File(ctx.getDir("bin", 0), SCRIPT_FILE);
        final ScriptRunner runner = new ScriptRunner(file, script, res, asroot);
        runner.start();
        try {
                if (timeout > 0) {
                        runner.join(timeout);
                } else {
                        runner.join();
                }
                if (runner.isAlive()) {
                        //设置超时
                        runner.interrupt();
                        runner.join(150);
                        runner.destroy();
                        runner.join(50);
                }
        } catch (InterruptedException ex) {
        }
        return runner.exitcode;
}
```

☑ 编写函数 runScriptAsRoot()，功能是在 Root 权限下执行脚本程序，具体代码如下所示。

```
public static int runScriptAsRoot(Context ctx, String script,
                StringBuilder res, long timeout) {
        return runScript(ctx, script, res, timeout, true);
}
```

☑ 编写函数 runScript()，功能是设置普通用户权限执行脚本程序，具体代码如下所示。

```
public static int runScript(Context ctx, String script, StringBuilder res)
            throws IOException {
        return runScript(ctx, script, res, 40000, false);
}
```

☑ 编写函数 assertBinaries()，功能是判断二进制文件在高速缓存目录是否被安装，具体代码如下所示。

```
public static boolean assertBinaries(Context ctx, boolean showErrors) {
        boolean changed = false;
        try {
            //检查 iptables_armv5 过滤包
            File file = new File(ctx.getDir("bin", 0), "iptables_armv5");
            if (!file.exists()) {
                copyRawFile(ctx, R.raw.iptables_armv5, file, "755");
                changed = true;
            }
            //检查 busybox
            file = new File(ctx.getDir("bin", 0), "busybox_g1");
            if (!file.exists()) {
                copyRawFile(ctx, R.raw.busybox_g1, file, "755");
                changed = true;
            }
            if (changed) {
                Toast.makeText(ctx, R.string.toast_bin_installed,
                        Toast.LENGTH_LONG).show();
            }
        } catch (Exception e) {
            if (showErrors)
                alert(ctx, "Error installing binary files: " + e);
            return false;
        }
        return true;
}
```

12.3.4　系统广播文件

编写文件 BootBroadcast.java，此文件是一个广播文件，在系统执行后将广播 ptables 规则。因为在规则中并没有设置开启显示信息，所以使用广播功能显示设置信息。文件 BootBroadcast.java 的主要代码如下所示。

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Handler;
import android.os.Message;
import android.widget.Toast;
public class BootBroadcast extends BroadcastReceiver {

    public void onReceive(final Context context, final Intent intent) {
        if (Intent.ACTION_BOOT_COMPLETED.equals(intent.getAction())) {
            if (Api.isEnabled(context)) {
                final Handler toaster = new Handler() {
```

```
                    public void handleMessage(Message msg) {
                        if (msg.arg1 != 0)
                            Toast.makeText(context, msg.arg1,
                                    Toast.LENGTH_SHORT).show();
                    }
                };
                //开启新线程阻止防火墙
                new Thread() {
                    @Override
                    public void run() {
                        if (!Api.applySavedIptablesRules(context, false)) {
                            // Error enabling firewall on boot
                            final Message msg = new Message();
                            msg.arg1 = R.string.toast_error_enabling;
                            toaster.sendMessage(msg);
                            Api.setEnabled(context, false);
                        }
                    }
                }.start();
            }
        }
    }
}
```

然后编写文件 PackageBroadcast.java，此文件也是一个具备广播功能的文件。当在手机中卸载一个软件后，会在防火墙中删除针对此软件的设置规则。文件 PackageBroadcast.java 的主要代码如下所示。

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class PackageBroadcast extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        if (Intent.ACTION_PACKAGE_REMOVED.equals(intent.getAction())) {
            //忽略应用更新
            final boolean replacing = intent.getBooleanExtra(Intent.EXTRA_REPLACING, false);
            if (!replacing) {
                final int uid = intent.getIntExtra(Intent.EXTRA_UID, -123);
                Api.applicationRemoved(context, uid);
            }
        }
    }
}
```

12.3.5　登录验证

编写文件 PassDialog.java，功能是在输入密码对话框中获取用户输入的密码，只有输入合法的密码数据才能登录系统。文件 PassDialog.java 的主要代码如下所示。

```
public class PassDialog extends Dialog implements android.view.View.OnClickListener,
```



```
android.view.View.OnKeyListener, OnCancelListener {
	private final Callback callback;
	private final EditText pass;
	/**创建一个对话框*/
	public PassDialog(Context context, boolean setting, Callback callback) {
		super(context);
		final View view = getLayoutInflater().inflate(R.layout.pass_dialog, null);
		((TextView)view.findViewById(R.id.pass_message)).setText(setting ? R.string.enternewpass :
R.string.enterpass);
		((Button)view.findViewById(R.id.pass_ok)).setOnClickListener(this);
		((Button)view.findViewById(R.id.pass_cancel)).setOnClickListener(this);
		this.callback = callback;
		this.pass = (EditText) view.findViewById(R.id.pass_input);
		this.pass.setOnKeyListener(this);
		setTitle(setting ? R.string.pass_titleset : R.string.pass_titleget);
		setOnCancelListener(this);
		setContentView(view);
	}
	@Override
	public void onClick(View v) {
		final Message msg = new Message();
		if (v.getId() == R.id.pass_ok) {
			msg.obj = this.pass.getText().toString();
		}
		dismiss();
		this.callback.handleMessage(msg);
	}
	@Override
	public boolean onKey(View v, int keyCode, KeyEvent event) {
		if (keyCode == KeyEvent.KEYCODE_ENTER) {
			final Message msg = new Message();
			msg.obj = this.pass.getText().toString();
			this.callback.handleMessage(msg);
			dismiss();
			return true;
		}
		return false;
	}
	@Override
	public void onCancel(DialogInterface dialog) {
		this.callback.handleMessage(new Message());
	}
}
```

12.3.6　打开/关闭某一个实施控件

编写文件 StatusWidget.java，功能是打开或关闭某一个实施控件，主要代码如下所示。

```
public class StatusWidget extends AppWidgetProvider {
```

```
    @Override
    public void onReceive(final Context context, final Intent intent) {
        super.onReceive(context, intent);
        if (Api.STATUS_CHANGED_MSG.equals(intent.getAction())) {
            //当防火墙状态改变时马上广播信息
            final Bundle extras = intent.getExtras();
            if (extras != null && extras.containsKey(Api.STATUS_EXTRA)) {
                final boolean firewallEnabled = extras
                        .getBoolean(Api.STATUS_EXTRA);
                final AppWidgetManager manager = AppWidgetManager
                        .getInstance(context);
                final int[ ] widgetIds = manager
                        .getAppWidgetIds(new ComponentName(context,
                                StatusWidget.class));
                showWidget(context, manager, widgetIds, firewallEnabled);
            }
        } else if (Api.TOGGLE_REQUEST_MSG.equals(intent.getAction())) {
            //根据防火墙开关信息广播状态信息
            final SharedPreferences prefs = context.getSharedPreferences(
                    Api.PREFS_NAME, 0);
            final boolean enabled = !prefs.getBoolean(Api.PREF_ENABLED, true);
            final String pwd = prefs.getString(Api.PREF_PASSWORD, "");
            if (!enabled && pwd.length() != 0) {
                Toast.makeText(context,
                        "Cannot disable firewall - password defined!",
                        Toast.LENGTH_SHORT).show();
                return;
            }
            final Handler toaster = new Handler() {
                public void handleMessage(Message msg) {
                    if (msg.arg1 != 0)
                        Toast.makeText(context, msg.arg1, Toast.LENGTH_SHORT)
                                .show();
                }
            };
            //开启新线程改变防火墙
            new Thread() {
                @Override
                public void run() {
                    final Message msg = new Message();
                    if (enabled) {
                        if (Api.applySavedIptablesRules(context, false)) {
                            msg.arg1 = R.string.toast_enabled;
                            toaster.sendMessage(msg);
                        } else {
                            msg.arg1 = R.string.toast_error_enabling;
                            toaster.sendMessage(msg);
                            return;
```

```
                        }
                    } else {
                        if (Api.purgeIptables(context, false)) {
                            msg.arg1 = R.string.toast_disabled;
                            toaster.sendMessage(msg);
                        } else {
                            msg.arg1 = R.string.toast_error_disabling;
                            toaster.sendMessage(msg);
                            return;
                        }
                    }
                    Api.setEnabled(context, enabled);
                }
            }.start();
        }
    }

    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager,
            int[ ] ints) {
        super.onUpdate(context, appWidgetManager, ints);
        final SharedPreferences prefs = context.getSharedPreferences(
                Api.PREFS_NAME, 0);
        boolean enabled = prefs.getBoolean(Api.PREF_ENABLED, true);
        showWidget(context, appWidgetManager, ints, enabled);
    }

    private void showWidget(Context context, AppWidgetManager manager,
            int[ ] widgetIds, boolean enabled) {
        final RemoteViews views = new RemoteViews(context.getPackageName(),
                R.layout.onoff_widget);
        final int iconId = enabled ? R.drawable.widget_on
                : R.drawable.widget_off;
        views.setImageViewResource(R.id.widgetCanvas, iconId);
        final Intent msg = new Intent(Api.TOGGLE_REQUEST_MSG);
        final PendingIntent intent = PendingIntent.getBroadcast(context, -1,
                msg, PendingIntent.FLAG_UPDATE_CURRENT);
        views.setOnClickPendingIntent(R.id.widgetCanvas, intent);
        manager.updateAppWidget(widgetIds, views);
    }
}
```

到此为止，整个网络流量防火墙系统介绍完毕。执行后的主界面效果如图 12-5 所示，按下 Menu 键后会弹出设置选项卡，如图 12-6 所示。

单击选项卡中的 Firewall disabled 按钮可以打开/关闭防火墙，单击选项卡中的 Log enabled 按钮可以打开/关闭日志，单击选项卡中的 Save rules 按钮会弹出保存进度条，如图 12-7 所示。

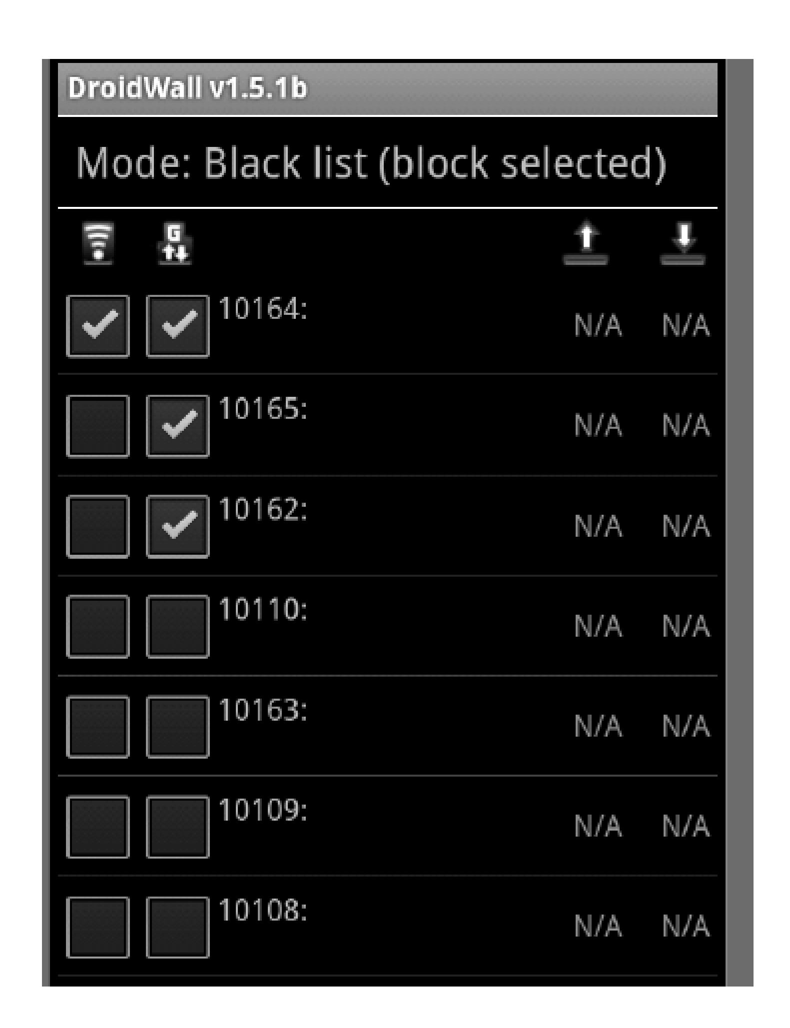


图 12-5　主界面

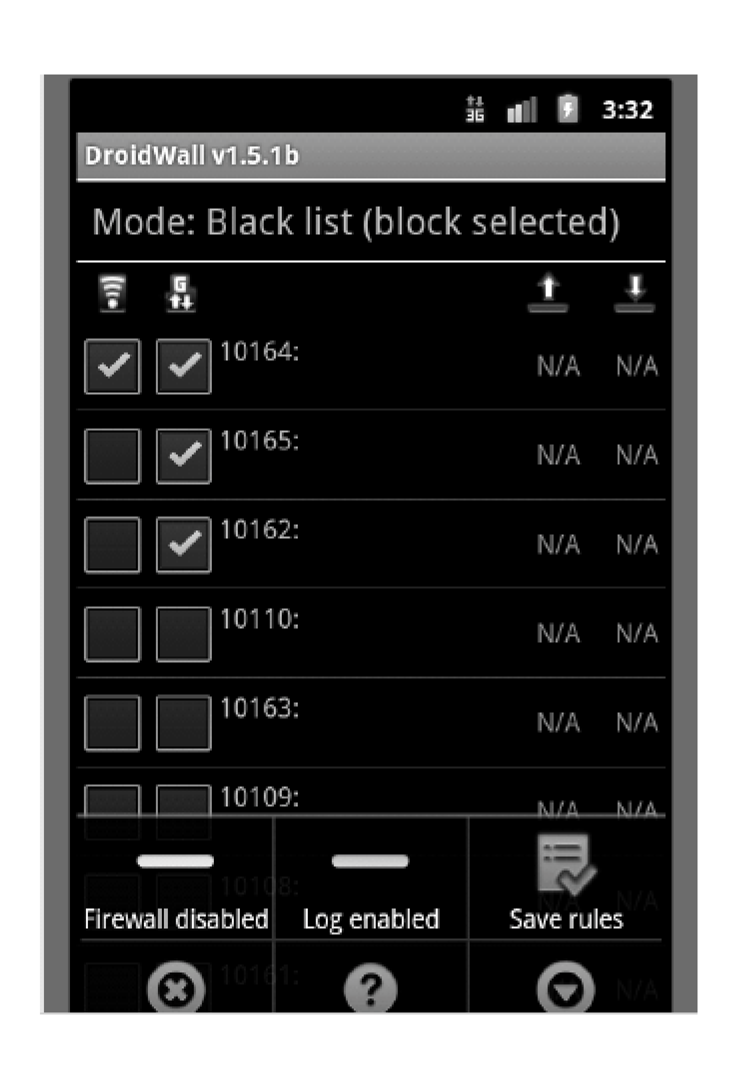


图 12-6　弹出设置选项卡

单击选项卡中的按钮会退出当前系统，单击选项卡中的按钮会弹出帮助对话框界面，如图 12-8 所示。



图 12-7　保存规则进度条

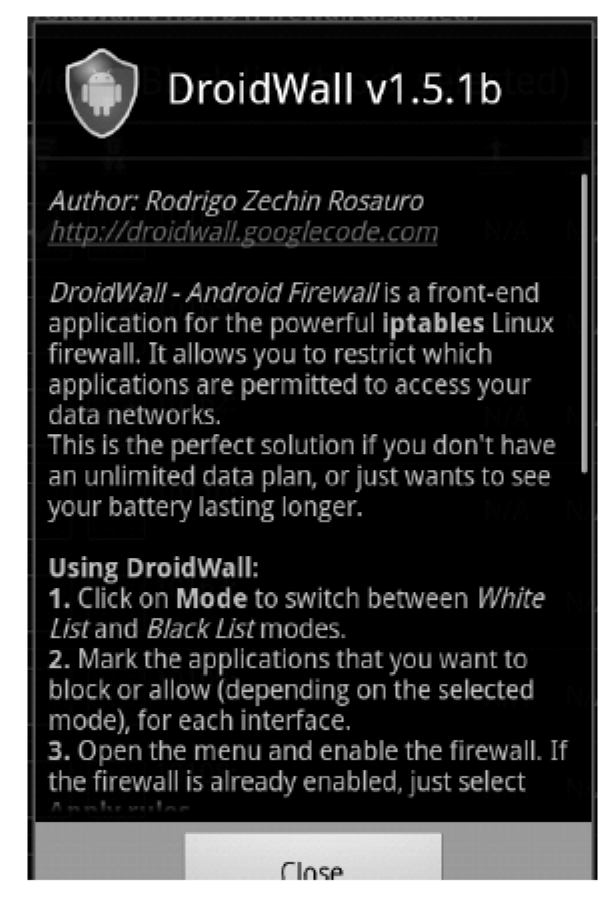


图 12-8　帮助对话框界面

单击选项卡中的按钮会弹出一个新的对话框，如图 12-9 所示。在此对话框中可以选择实现其他功能，例如，选择 Set password 选项后会弹出一个设置密码界面，如图 12-10 所示。



图 12-9　新功能对话框

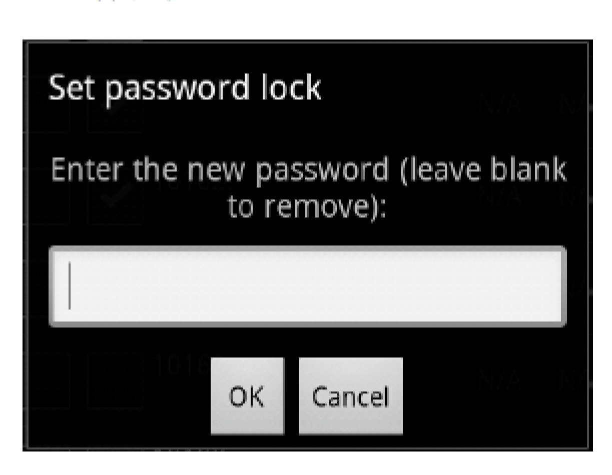


图 12-10　设置密码界面



第 13 章　Map 地图

Android 提供的地图（Map）功能可能是广大开发者非常关心的一个部分。到目前为止，开发内嵌式地图应用的软件相当困难，而且往往还需要支付很高的地图厂商的版权费用，加之手机上 GPS 功能的不完善，导致很多可以基于当前位置来开发功能的软件少之又少。本章将详细介绍 Map 在 Android 中的应用，并通过一个综合实例的实现过程来讲解具体实现流程。

13.1　项 目 分 析

知识点讲解：光盘：视频\视频讲解\第 13 章\项目分析.avi

本项目实例的功能是，为用户提供需要的目标定位处理，即用户设置一个目标后，可以在后台启动一个 Service，能够定时读取 GPS 数据以获得用户目前所在的位置信息，并将其保存在数据库中。用户可以选择其他目标信息，也能够将这些轨迹显示在 Map 地图上。本项目的实现流程如图 13-1 所示。



图 13-1　实现流程

13.1.1　规划 UI 界面

本项目 UI 界面的结构如图 13-2 所示。

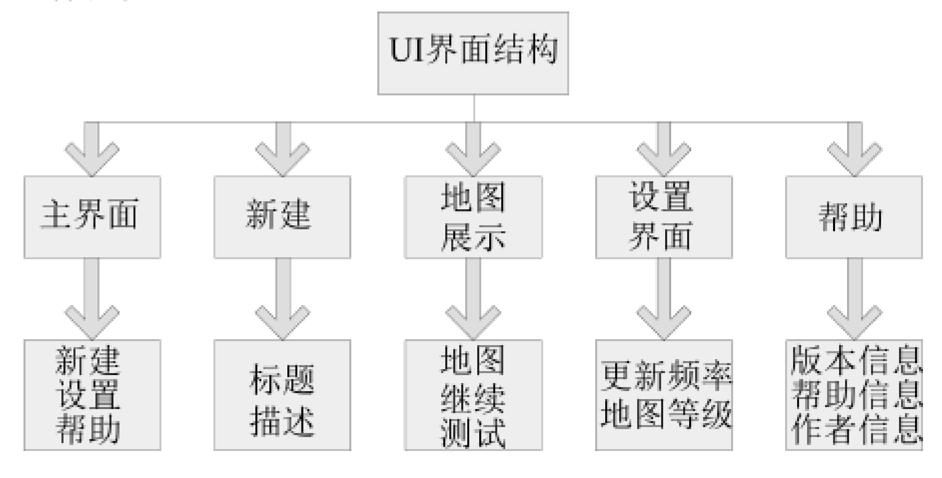


图 13-2　UI 界面结构

13.1.2　数据存储设计

数据存储既可以通过文件系统实现，也可以通过专用数据库工具实现。但是为了保证系统的日后维护

工作，本项目使用数据库工具方式，选用的是最常见的 SQlite。

根据前面介绍的系统需求分析，本系统用到了 3 类数据，一类是目标名，一类是每次追踪时的位置信息，另外一类是配置信息。为此，在本系统需要设计两个表来存储数据，具体说明如下。

表 Tracks 用于存储目标信息，具体结构如表 13-1 所示。

表 13-1 Tracks 结构

属　性	类　型	说　明
id	INTEGER	主键
name	TEXT	名称
desc	TEXT	说明
distance	LONG	距离
tracked_time	LONG	时间
locates_count	INTEGER	点数
created_at	INTEGER	创建时间
update_at	INTEGER	更新时间
avg_speed	LONG	平均速度
max_speed	LONG	最大速度

表 Locats 用于存储目标的位置信息，具体结构如表 13-2 所示。

表 13-2 Locats 结构

属　性	类　型	说　明
id	INTEGER	主键
track_id	INTEGER	跟踪的目标 ID
longitude	TEXT	纬度
latiude	TEXT	经度
altitude	TEXT	偏差
created_at	INTEGER	创建时间

13.2 具 体 实 现

知识点讲解：光盘：视频\视频讲解\第 13 章\具体实现.avi

项目的准备工作做好后，接下来将介绍本项目的具体实现过程，希望读者认真体会每一段代码的功能和编写原理，为提高自己的开发水平做准备。

13.2.1 新建工程

打开 Eclipse，依次选择 File | New | Android Project 命令，新建一个名为 map 的工程文件，如图 13-3 所示。

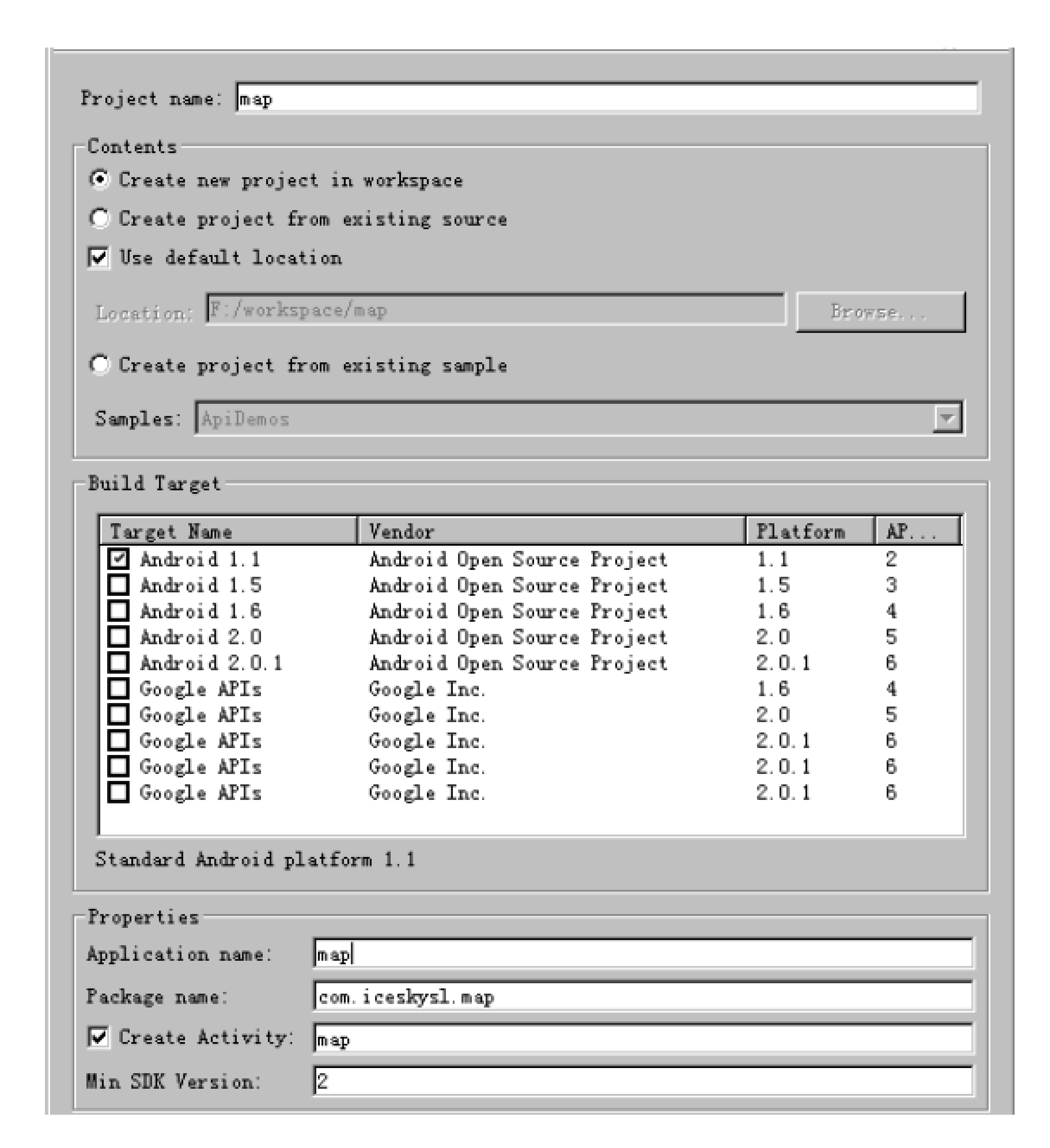


图 13-3　新建工程

13.2.2　主界面

主界面即项目执行后首先显示的界面，具体流程如下。

（1）编写主布局文件 main.xml，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/title"
    />

     <ListView android:id="@id/android:list"
          android:layout_width="fill_parent"
           android:layout_height="wrap_content"
           android:drawSelectorOnTop="false" />

     <TextView android:id="@+id/android:empty"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="@string/start" />
</LinearLayout>
```

（2）编写一个“历史记录”的列表信息，只显示系统数据库内的前 10 条数据。具体功能是在文件 string.xml 中实现的，具体代码如下所示。

```
<string name="title">历史记录:</string>
<string name="app_name">aaa</string>
<string name="app_title">bbbb</string>
<string name="menu_main">主页</string>
<string name="menu_new">新建</string>
<string name="menu_con">继续</string>
<string name="menu_del">删除</string>
<string name="menu_setting">设置</string>
<string name="menu_helps">帮助</string>
<string name="menu_back">返回</string>
<string name="menu_exit">退出</string>
```

（3）编写 onCreate()方法，具体代码如下所示。

```
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
       render_tracks();
}
```

在上述代码中，需要将以往的历史记录从数据库中读取出来，显示在列表中，然后使用 render_tracks()方法将数据库的历史记录读取出来，并更新到列表中。

（4）在 iTracks.java 中编写实现菜单的代码，主要代码如下所示。

```
//定义菜单需要的常量
private static final int MENU_NEW = Menu.FIRST + 1;
private static final int MENU_CON = MENU_NEW + 1;
private static final int MENU_SETTING = MENU_CON + 1;
private static final int MENU_HELPS = MENU_SETTING + 1;
private static final int MENU_EXIT = MENU_HELPS + 1;
//初始化菜单
@Override
public boolean onCreateOptionsMenu(Menu menu) {
       Log.d(TAG, "onCreateOptionsMenu");

       super.onCreateOptionsMenu(menu);

       menu.add(0, MENU_NEW, 0, R.string.menu_new).setIcon(
                   R.drawable.new_track).setAlphabeticShortcut('N');
       menu.add(0, MENU_CON, 0, R.string.menu_con).setIcon(
                   R.drawable.con_track).setAlphabeticShortcut('C');
       menu.add(0, MENU_SETTING, 0, R.string.menu_setting).setIcon(
                   R.drawable.setting).setAlphabeticShortcut('S');
       menu.add(0, MENU_HELPS, 0, R.string.menu_helps).setIcon(
                   R.drawable.helps).setAlphabeticShortcut('H');
       menu.add(0, MENU_EXIT, 0, R.string.menu_exit).setIcon(
                   R.drawable.exit).setAlphabeticShortcut('E');
       return true;
}
```

```
//当一个菜单被选中时调用
@Override
public boolean onOptionsItemSelected(MenuItem item) {
	Intent intent = new Intent();
	switch (item.getItemId()) {
	case MENU_NEW:
		intent.setClass(iTracks.this, NewTrack.class);
		startActivity(intent);
		return true;
	case MENU_CON:
		//TODO: 继续跟踪选择的记录
		conTrackService();
		return true;
	case MENU_SETTING:
		intent.setClass(iTracks.this, Setting.class);
		startActivity(intent);
		return true;
	case MENU_HELPS:
		intent.setClass(iTracks.this, Helps.class);
		startActivity(intent);
		return true;
	case MENU_EXIT:
		finish();
		break;
	}
	return true;
}
```

通过上述代码，创建了菜单框架和菜单被选中后的响应方法。至此，主界面的主要代码介绍完毕，执行后的效果如图 13-4 所示。



图 13-4　主界面

13.2.3　新建界面

当在图 13-4 中单击“新建”按钮后，会弹出新建目标记录界面，实现该界面的具体流程如下。

（1）编写布局文件 new_track.xml，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <TextView android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="@string/new_tips" />
        <TextView android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="@string/new_name" />
        <EditText android:id="@+id/new_name"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="" />
        <TextView android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="@string/new_desc" />
        <EditText android:id="@+id/new_desc"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:scrollbars="vertical"/>
        <Button android:id="@+id/new_submit"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/new_submit" />
</LinearLayout>
```

在上述代码中，用 TextView 来显示提示信息，用 EditText 来接收用户的输入。

（2）编写处理文件 NewTrack.java，具体代码如下所示。

```
package com.iceskysl.map;


import com.iceskysl.map.R;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class NewTrack extends Activity {
       private static final String TAG = "NewTrack";
        private Button button_new;
        private EditText field_new_name;
        private EditText field_new_desc;
```



```
    private TrackDbAdapter mDbHelper;

    /** Called when the activity is first created */
   @Override
   public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.new_track);
         setTitle(R.string.menu_new);
           findViews();
           setListensers();

           mDbHelper = new TrackDbAdapter(this);
           mDbHelper.open();
   }

    @Override
    protected void onStop(){
           super.onStop();
           Log.d(TAG, "onStop");
           mDbHelper.close();
    }

    private void findViews() {
           Log.d(TAG, "find Views");
           field_new_name = (EditText) findViewById(R.id.new_name);
           field_new_desc = (EditText) findViewById(R.id.new_desc);
           button_new = (Button) findViewById(R.id.new_submit);
    }

    // Listen for button clicks
    private void setListensers() {
           Log.d(TAG, "set Listensers");
           button_new.setOnClickListener(new_track);
    }

    private Button.OnClickListener new_track = new Button.OnClickListener() {
           public void onClick(View v) {
                  Log.d(TAG, "onClick new_track...");
                  try {
                         String name = (field_new_name.getText().toString());
                         String desc = (field_new_desc.getText()
                                        .toString());
                         if (name.equals("")) {
                                Toast.makeText(NewTrack.this,
                                              getString(R.string.new_name_null),
                                              Toast.LENGTH_SHORT).show();
                         } else {
                                // TODO 调用存储接口保存到数据库并启动 Service
                                Long row_id = mDbHelper.createTrack(name, desc);
                                Log.d(TAG, "row_id="+row_id);
```

```
                    Intent intent = new Intent();
                    intent.setClass(NewTrack.this, ShowTrack.class);
                    intent.putExtra(TrackDbAdapter.KEY_ROWID, row_id);
                    intent.putExtra(TrackDbAdapter.NAME, name);
                    intent.putExtra(TrackDbAdapter.DESC, desc);

                    startActivity(intent);
                }
            } catch (Exception err) {
                Log.e(TAG, "error: " + err.toString());
                Toast.makeText(NewTrack.this, getString(R.string.new_fail),
                        Toast.LENGTH_SHORT).show();
            }
        }
    };
}
```

在上述代码中，首先在方法 onCreate()中设置其关联的 layout；然后调用 findViewsByid()来获取名字和 EditText 组件，并获取提交按钮；最后，定义了一个 Button.OnClickListener new_track 对象，实现其 onClick()方法。

至此，本模块的主要功能设计完毕，执行后的效果如图 13-5 所示。



图 13-5　新建界面

13.2.4　设置界面

当在图 13-4 中单击“设置”按钮后，会弹出系统设置界面，实现该界面的具体流程如下。

（1）编写布局文件 setting.xml，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:id="@+id/setting_tips"
        android:layout_width="fill_parent"
```

```
            android:layout_height="wrap_content"
            android:text="" />
        <TextView android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/setting_gps" />
                <Spinner android:id="@+id/setting_gps"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:drawSelectorOnTop="true"
          android:prompt="@string/spinner_gps_prompt"
      />
        <TextView android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/setting_map_level" />
      <Spinner android:id="@+id/setting_map_level"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:drawSelectorOnTop="true"
          android:prompt="@string/spinner_map_prompt"
      />
        <Button android:id="@+id/setting_submit"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/setting_submit" />
</LinearLayout>
```

在上述代码中，通过 Spinner 组件实现了一个供用户使用的下拉菜单。

（2）编写处理文件 Setting.java，具体代码如下所示。

```
package com.iceskysl.map;

import com.iceskysl.map.R;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.Spinner;
import android.widget.Toast;

public class Setting extends Activity {
      private static final String TAG = "Setting";
       //定义菜单需要的常量
       private static final int MENU_MAIN = Menu.FIRST + 1;
       private static final int MENU_NEW = MENU_MAIN + 1;
       private static final int MENU_BACK = MENU_NEW + 1;
```

```
    //保存个性化设置
    public static final String SETTING_INFOS = "SETTING_Infos";
    public static final String SETTING_GPS = "SETTING_Gps";
    public static final String SETTING_MAP = "SETTING_Map";
    public static final String SETTING_GPS_POSITON = "SETTING_Gps_p";
    public static final String SETTING_MAP_POSITON = "SETTING_Map_p";

    private Button button_setting_submit;
    private Spinner field_setting_gps;
    private Spinner field_setting_map_level;


   @Override
   public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.setting);
         setTitle(R.string.menu_setting);
           findViews();
           setListensers();
           restorePrefs();
   }

    private void findViews() {
           Log.d(TAG, "find Views");
           button_setting_submit = (Button) findViewById(R.id.setting_submit);
           field_setting_gps = (Spinner) findViewById(R.id.setting_gps);
         ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
                     this, R.array.gps, android.R.layout.simple_spinner_item);
         adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
         field_setting_gps.setAdapter(adapter);

           field_setting_map_level = (Spinner) findViewById(R.id.setting_map_level);
         ArrayAdapter<CharSequence> adapter2 = ArrayAdapter.createFromResource(
                     this, R.array.map, android.R.layout.simple_spinner_item);
         adapter2.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
         field_setting_map_level.setAdapter(adapter2);
    }

    // Listen for button clicks
    private void setListensers() {
           Log.d(TAG, "set Listensers");
           button_setting_submit.setOnClickListener(setting_submit);
    }
    private Button.OnClickListener setting_submit = new Button.OnClickListener() {
           public void onClick(View v) {
                  Log.d(TAG, "onClick new_track..");
                  try {
                         String gps = (field_setting_gps.getSelectedItem().toString());
                         String map = (field_setting_map_level.getSelectedItem()
                                       .toString());
```

```
                if (gps.equals("") || map.equals("")) {
                    Toast.makeText(Setting.this,
                            getString(R.string.setting_null),
                            Toast.LENGTH_SHORT).show();
                } else {
                    //保存设定
                    storePrefs();
                    Toast.makeText(Setting.this,
                            getString(R.string.setting_ok),
                            Toast.LENGTH_SHORT).show();
                    //跳转到主界面
                    Intent intent = new Intent();
                    intent.setClass(Setting.this, iTracks.class);
                    startActivity(intent);
                }
            } catch (Exception err) {
                Log.e(TAG, "error: " + err.toString());
                Toast.makeText(Setting.this, getString(R.string.setting_fail),
                        Toast.LENGTH_SHORT).show();
            }
        }
    };

    //Restore preferences
    private void restorePrefs() {
        SharedPreferences settings = getSharedPreferences(SETTING_INFOS, 0);
        int setting_gps_p = settings.getInt(SETTING_GPS_POSITON, 0);
        int setting_map_level_p = settings.getInt(SETTING_MAP_POSITON, 0);
        Log.d(TAG, "restorePrefs: setting_gps= "+ setting_gps_p + ",setting_map_level=" + setting_map_level_p);

        if (setting_gps_p != 0 && setting_map_level_p != 0) {
            field_setting_gps.setSelection(setting_gps_p);
            field_setting_map_level.setSelection(setting_map_level_p);
            button_setting_submit.requestFocus();
        }else if(setting_gps_p != 0 ){
            field_setting_gps.setSelection(setting_gps_p);
            field_setting_map_level.requestFocus();
        }else if(setting_map_level_p != 0){
            field_setting_map_level.setSelection(setting_map_level_p);
            field_setting_gps.requestFocus();
        }else{
            field_setting_gps.requestFocus();
        }
    }

    @Override
    protected void onStop(){
        super.onStop();
        Log.d(TAG, "save setting infos");
        // Save user preferences. We need an Editor object to
        // make changes. All objects are from android.context.Context
```

```
            storePrefs();
        }

        //保存个人设置
        private void storePrefs() {
            Log.d(TAG, "storePrefs setting infos");
            SharedPreferences settings = getSharedPreferences(SETTING_INFOS, 0);
            settings.edit()
                .putString(SETTING_GPS, field_setting_gps.getSelectedItem().toString())
                .putString(SETTING_MAP, field_setting_map_level.getSelectedItem().toString())
                .putInt(SETTING_GPS_POSITON, field_setting_gps.getSelectedItemPosition())
                .putInt(SETTING_MAP_POSITON, field_setting_map_level.getSelectedItemPosition())
                .commit();
        }

        //初始化菜单
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            super.onCreateOptionsMenu(menu);
            menu.add(0, MENU_MAIN, 0, R.string.menu_main).setIcon(
                    R.drawable.icon).setAlphabeticShortcut('M');
            menu.add(0, MENU_NEW, 0, R.string.menu_new).setIcon(
                    R.drawable.new_track).setAlphabeticShortcut('N');
            menu.add(0, MENU_BACK, 0, R.string.menu_back).setIcon(
                    R.drawable.back).setAlphabeticShortcut('E');
            return true;
        }

        //当一个菜单被选中时调用
        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            Intent intent = new Intent();
            switch (item.getItemId()) {
            case MENU_NEW:
                intent.setClass(Setting.this, NewTrack.class);
                startActivity(intent);
                return true;
            case MENU_MAIN:
                intent.setClass(Setting.this, iTracks.class);
                startActivity(intent);
                return true;
            case MENU_BACK:
                finish();
                break;
            }
            return true;
        }

}
```

上述代码的具体实现流程如下。

第 1 步：声明需要的变量。

第 2 步：在 onCreat()中绑定 setting.xml 为其布局模板。

第 3 步：使用 setContentView()设定其对应的布局文件 setting.xml，使用 setTitle()设定其标题，进一步调用 findViews()查询需要的操作组件，并调用 setListensers()给按钮设定单击监听器，最后调用 restorePrefs()将默认值或用户的历史选择值显示出来。

第 4 步：用 findViews()找到需要用到的组件。

其中，下拉列表框中的内容是预先设置好的，定义在 array.xml 中，具体代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- Used in View/setting.java -->
    <string-array name="gps">
        <item>1</item>
        <item>10</item>
        <item>20</item>
        <item>30</item>
        <item>40</item>
        <item>50</item>
    </string-array>
    <string-array name="map">
        <item>2</item>
        <item>3</item>
        <item>4</item>
        <item>5</item>
        <item>20</item>
        <item>30</item>
        <item>41</item>
        <item>52</item>
        <item>63</item>
        <item>74</item>
        <item>85</item>
        <item>96</item>
    </string-array>
</resources>
```

至此，本模块的主要代码介绍完毕，执行后的效果如图 13-6 所示。



图 13-6　设置界面

13.2.5　帮助界面

在图 13-4 中单击“帮助”按钮，弹出系统默认的帮助界面，实现该界面的具体流程如下。

（1）编写布局文件 helps.xml，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      >
<TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/version"
      />
<TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/version_text"
      />
<TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/helps_infos"
      />
     <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:autoLink="all"
      android:text="@string/helps_text"
 />
   <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/author"
      />
      <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:autoLink="all"
      android:text="@string/author_text"
 />
</LinearLayout>
```

在上述代码中，通过 TextView 显示了各条帮助信息。

（2）编写处理文件 helps.java，主要代码如下所示。

```
package com.iceskysl.map;

import com.iceskysl.map.R;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
```

```
import android.view.MenuItem;

public class Helps extends Activity {
	//定义菜单需要的常量
	private static final int MENU_MAIN = Menu.FIRST + 1;
	private static final int MENU_NEW = MENU_MAIN + 1;
	private static final int MENU_BACK = MENU_NEW + 1;
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.helps);
	setTitle(R.string.menu_helps);
	}
	//初始化菜单
	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		super.onCreateOptionsMenu(menu);
		menu.add(0, MENU_MAIN, 0, R.string.menu_main).setIcon(
				R.drawable.icon).setAlphabeticShortcut('M');
		menu.add(0, MENU_NEW, 0, R.string.menu_new).setIcon(
				R.drawable.new_track).setAlphabeticShortcut('N');
		menu.add(0, MENU_BACK, 0, R.string.menu_back).setIcon(
				R.drawable.back).setAlphabeticShortcut('E');
		return true;
	}

	//当一个菜单被选中时调用
	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		Intent intent = new Intent();
		switch (item.getItemId()) {
		case MENU_NEW:
			intent.setClass(Helps.this, NewTrack.class);
			startActivity(intent);
			return true;
		case MENU_MAIN:
			intent.setClass(Helps.this, iTracks.class);
			startActivity(intent);
			return true;
		case MENU_BACK:
			finish();
			break;
		}
		return true;
	}
}
```

在上述代码中，首先在 onCreate()方法中设定其对应的布局文件为 helps.xml，然后添加菜单和菜单对应的功能。

至此，本模块的主要代码介绍完毕，执行后的效果如图 13-7 所示。



图 13-7 帮助界面

13.2.6 地图界面

前面介绍的都是主菜单中的选项，下面开始讲解比较复杂的功能：将地图在 Android 手机中显示。通过 android.maps 中的 MapView 即可方便地实现编程工作。基本实现流程如下所示。

1. 申请 APIKey

（1）打开 Eclipse，依次选择 Windows | Preferences | Android | Build 命令，查看默认的 debug.keystore 位置，如图 13-8 所示。

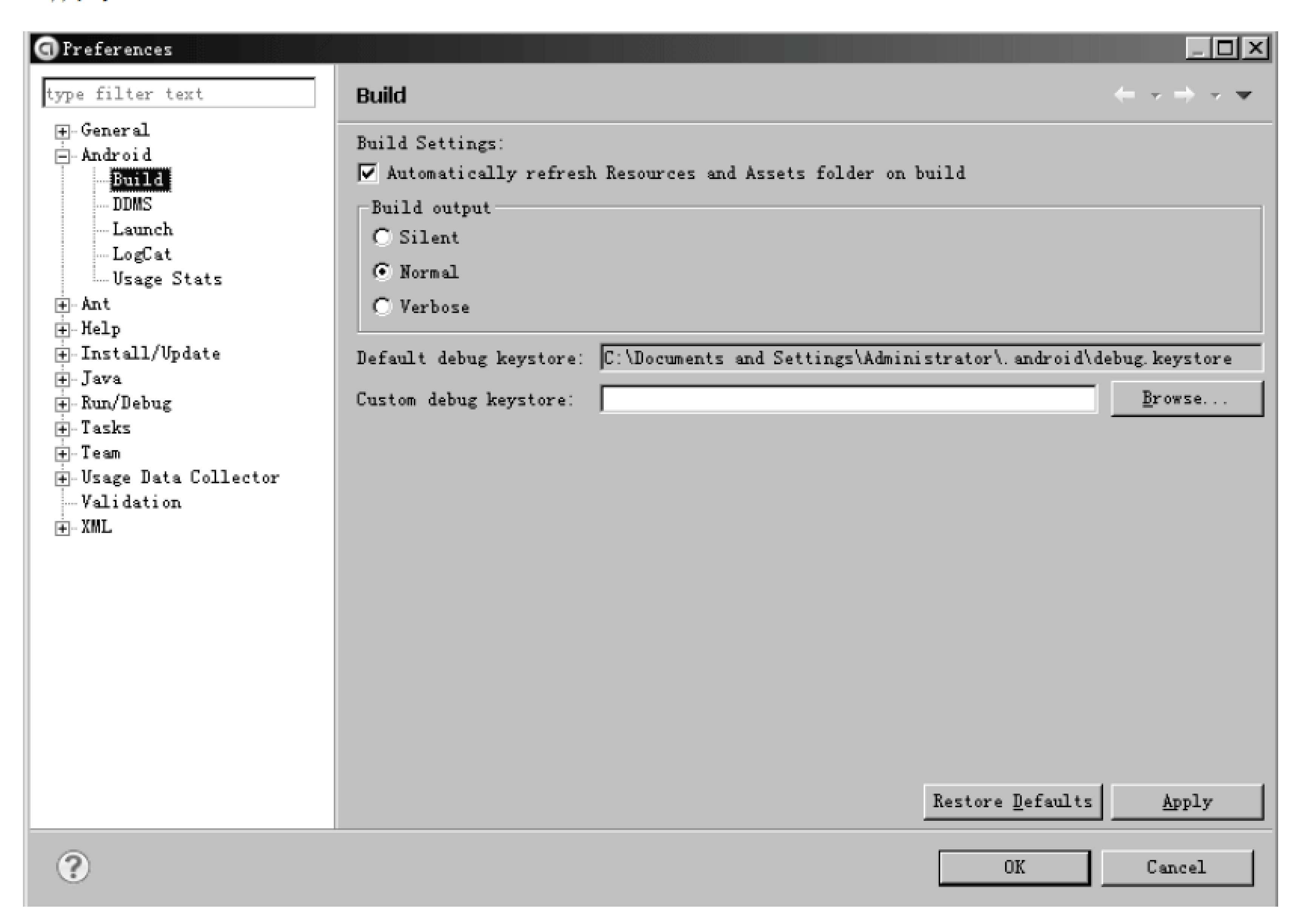


图 13-8 debug keystore 位置

（2）打开 cmd，在 cmd 中执行如下命令。



```
keytool -list -alias androiddebugkey -keystore "C:\Documents and Settings\Administrator\.android\debug.keystore" -storepass android -keypass android
```

通过上述命令获取 MD5 指纹，此时系统会提示输入 keystore 代码，输入 android 后会显示要获取的指纹。

（3）打开网址，输入得到的 MD5 指纹，然后单击 Generate API Key 按钮申请获取 APIKey，如图 13-9 所示。

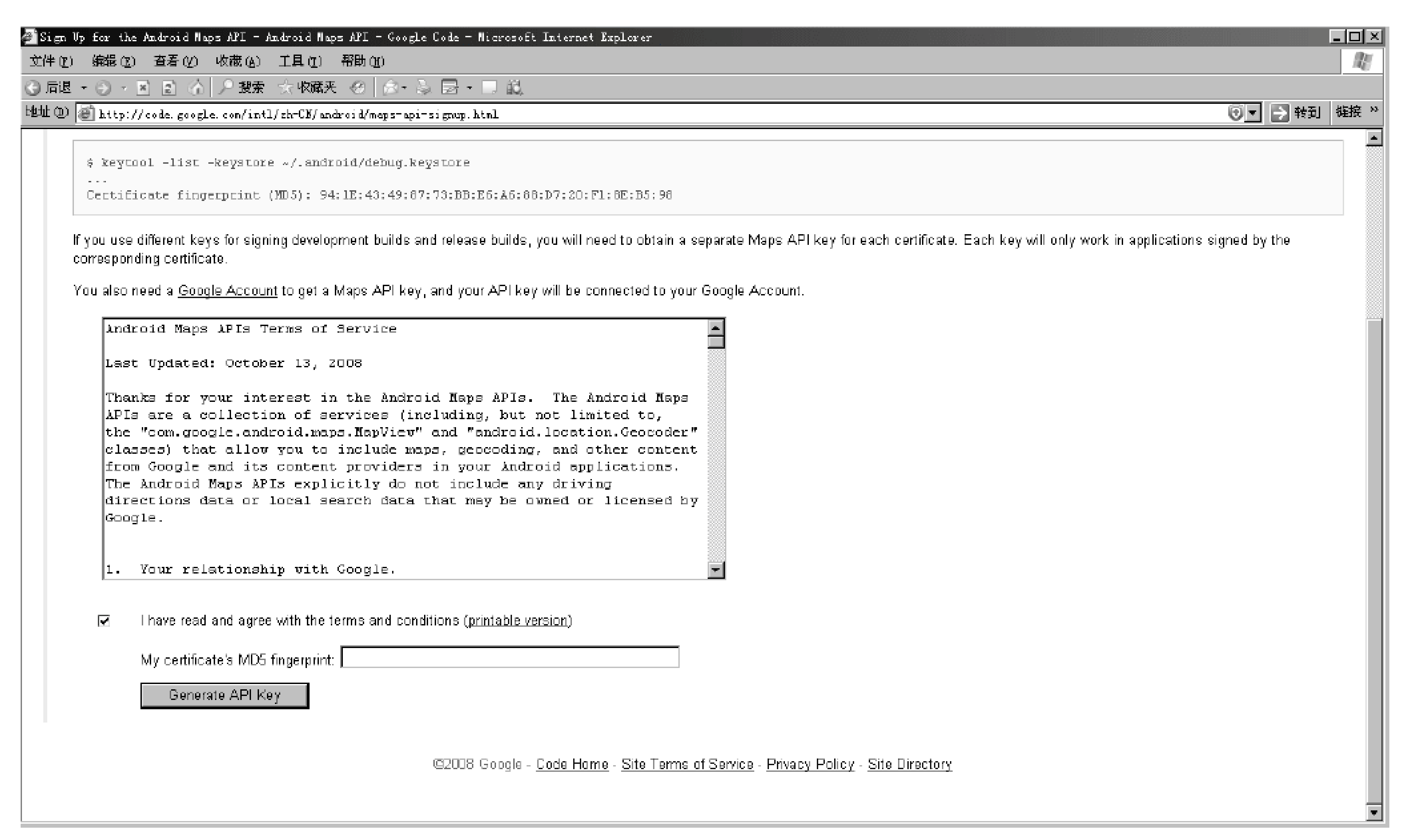


图 13-9　获取 APIKey

2. 编码

下面介绍编码过程，具体流程如下。

（1）编写布局文件 show_track.xml，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<view android:id="@+id/mv"
    class="com.xxx.android.maps.MapView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1"
    android:apiKey="01Yu9W3X3vbr4UFCa_OlwALuXpyD__ocgLaiYNw"
    />
<LinearLayout xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
```

```
    android:background="#550000ff"
    android:padding="1px"
    >
<Button android:id="@+id/sat"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="40px"
    android:text="@string/satellite" />
<Button android:id="@+id/traffic"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="@string/traffic" />
<Button android:id="@+id/streetview"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="@string/street" />

<Button android:id="@+id/gps"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="GPS" />
</LinearLayout>
<LinearLayout xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="fill_parent"
    android:background="#550000ff"
    android:padding="1px"
    >
<Button android:id="@+id/zin"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="30px"
    android:text="+"/>
<Button android:id="@+id/zout"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="-"/>
<Button android:id="@+id/pann"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="N"/>
<Button android:id="@+id/pane"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="E"/>

<Button android:id="@+id/panw"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
```

```
        android:text="W"/>
<Button android:id="@+id/pans"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="S"/>
</LinearLayout>
</FrameLayout>
```

通过上述代码，用 MapView 组件来显示地图，并通过设置的按钮来控制地图，例如放大、缩小、移动和模式的转换。

（2）编写处理文件 ShowTrack.java，具体代码如下所示。

```
package com.iceskysl.map;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.Resources;
import android.database.Cursor;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Point;
import android.graphics.RectF;
import android.graphics.Paint.Style;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.util.Log;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

import com.xxx.android.maps.GeoPoint;
import com.xxx.android.maps.MapActivity;
import com.xxx.android.maps.MapController;
import com.xxx.android.maps.MapView;
import com.xxx.android.maps.MyLocationOverlay;
import com.xxx.android.maps.Overlay;
import com.iceskysl.map.R;

public class ShowTrack extends MapActivity {
    //定义菜单需要的常量
    private static final int MENU_NEW = Menu.FIRST + 1;
    private static final int MENU_CON = MENU_NEW + 1;
    private static final int MENU_DEL = MENU_CON + 1;
    private static final int MENU_MAIN = MENU_DEL + 1;
```

```
private TrackDbAdapter mDbHelper;
private LocateDbAdapter mlcDbHelper;

private static final String TAG = "ShowTrack";
private static MapView mMapView;
private MapController mc;

protected MyLocationOverlay mOverlayController;
private Button mZin;
private Button mZout;
private Button mPanN;
private Button mPanE;
private Button mPanW;
private Button mPanS;
private Button mGps;
private Button mSat;
private Button mTraffic;
private Button mStreetview;
private String mDefCaption = "";
private GeoPoint mDefPoint;

private LocationManager lm;
private LocationListener locationListener;

private int track_id;
private Long rowId;

/** Called when the activity is first created*/
public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.show_track);
        findViews();
        centerOnGPSPosition();
        revArgs();
        // //////////////
          //mDbHelper = new TrackDbAdapter(this);
          //mDbHelper.open();

        //mlcDbHelper = Track.getDbHelp();
                //new LocateDbAdapter(this);
        //mlcDbHelper.open();

        paintLocates();
        startTrackService();
}

private void startTrackService() {
        Intent i = new Intent("com.iceskysl.iTracks.START_TRACK_SERVICE");
        i.putExtra(LocateDbAdapter.TRACKID, track_id);
      startService(i);
}
```

```
private void stopTrackService() {
    stopService(new Intent("com.iceskysl.iTracks.START_TRACK_SERVICE"));
}

private void paintLocates() {
        mlcDbHelper = new LocateDbAdapter(this);
        mlcDbHelper.open();
        Cursor mLocatesCursor = mlcDbHelper.getTrackAllLocates(track_id);
        startManagingCursor(mLocatesCursor);
        Resources resources = getResources();
        Overlay overlays = new LocateOverLay(resources
                        .getDrawable(R.drawable.icon), mLocatesCursor);
        mMapView.getOverlays().add(overlays);
        mlcDbHelper.close();
}

private void revArgs() {
        Log.d(TAG, "revArgs.");
        Bundle extras = getIntent().getExtras();
        if (extras != null) {
                String name = extras.getString(TrackDbAdapter.NAME);
                //String desc = extras.getString(TrackDbAdapter.DESC);
                rowId = extras.getLong(TrackDbAdapter.KEY_ROWID);
                track_id = rowId.intValue();
                Log.d(TAG, "rowId=" + rowId);
                if (name != null) {
                        setTitle(name);
                }
        }
}

protected boolean isRouteDisplayed() {
        // TODO Auto-generated method stub
        return false;
}

private void findViews() {
        Log.d(TAG, "find Views");
        //Get the map view from resource file
        mMapView = (MapView) findViewById(R.id.mv);
        mc = mMapView.getController();

        SharedPreferences settings = getSharedPreferences(Setting.SETTING_INFOS, 0);
        String setting_gps = settings.getString(Setting.SETTING_MAP, "10");
        mc.setZoom(Integer.parseInt(setting_gps));

        //Set up the button for "Pan East"
        mPanE = (Button) findViewById(R.id.sat);
        mPanE.setOnClickListener(new OnClickListener() {
                // @Override
```

```
        public void onClick(View arg0) {
            panEast();
        }
    });
    //Set up the button for "Zoom In"
    mZin = (Button) findViewById(R.id.zin);
    mZin.setOnClickListener(new OnClickListener() {
        // @Override
        public void onClick(View arg0) {
            zoomIn();
        }
    });
    //Set up the button for "Zoom Out"
    mZout = (Button) findViewById(R.id.zout);
    mZout.setOnClickListener(new OnClickListener() {
        // @Override
        public void onClick(View arg0) {
            zoomOut();
        }
    });
    //Set up the button for "Pan North"
    mPanN = (Button) findViewById(R.id.pann);
    mPanN.setOnClickListener(new OnClickListener() {
        // @Override
        public void onClick(View arg0) {
            panNorth();
        }
    });

    // Set up the button for "Pan East"
    mPanE = (Button) findViewById(R.id.pane);
    mPanE.setOnClickListener(new OnClickListener() {
        // @Override
        public void onClick(View arg0) {
            panEast();
        }
    });

    // Set up the button for "Pan West"
    mPanW = (Button) findViewById(R.id.panw);
    mPanW.setOnClickListener(new OnClickListener() {
        // @Override
        public void onClick(View arg0) {
            panWest();
        }
    });
    // Set up the button for "Pan South"
    mPanS = (Button) findViewById(R.id.pans);
    mPanS.setOnClickListener(new OnClickListener() {
        // @Override
        public void onClick(View arg0) {
```

```
                panSouth();
            }
        });

        // Set up the button for "GPS"
        mGps = (Button) findViewById(R.id.gps);
        mGps.setOnClickListener(new OnClickListener() {
            // @Override
            public void onClick(View arg0) {
                centerOnGPSPosition();
            }
        });
        // Set up the button for "Satellite toggle"
        mSat = (Button) findViewById(R.id.sat);
        mSat.setOnClickListener(new OnClickListener() {
            // @Override
            public void onClick(View arg0) {
                toggleSatellite();
            }
        });

        // Set up the button for "Traffic toggle"
        mTraffic = (Button) findViewById(R.id.traffic);
        mTraffic.setOnClickListener(new OnClickListener() {
            // @Override
            public void onClick(View arg0) {
                toggleTraffic();
            }
        });

        // Set up the button for "Traffic toggle"
        mStreetview = (Button) findViewById(R.id.streetview);
        mStreetview.setOnClickListener(new OnClickListener() {
            // @Override
            public void onClick(View arg0) {
                toggleStreetView();
            }
        });

        // ---use the LocationManager class to obtain GPS locations---
        lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
        locationListener = new MyLocationListener();
        lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
                locationListener);
    }

    public boolean onKeyDown(int keyCode, KeyEvent event) {
        Log.d(TAG, "onKeyDown");
        if (keyCode == KeyEvent.KEYCODE_DPAD_LEFT) {
            panWest();
            return true;
```

```
        } else if (keyCode == KeyEvent.KEYCODE_DPAD_RIGHT) {
            panEast();
            return true;
        } else if (keyCode == KeyEvent.KEYCODE_DPAD_UP) {
            panNorth();
            return true;
        } else if (keyCode == KeyEvent.KEYCODE_DPAD_DOWN) {
            panSouth();
            return true;
        }
        return false;
    }

    public void panWest() {
        GeoPoint pt = new GeoPoint(mMapView.getMapCenter().getLatitudeE6(),
                mMapView.getMapCenter().getLongitudeE6()
                        - mMapView.getLongitudeSpan() / 4);
        mc.setCenter(pt);
    }

    public void panEast() {
        GeoPoint pt = new GeoPoint(mMapView.getMapCenter().getLatitudeE6(),
                mMapView.getMapCenter().getLongitudeE6()
                        + mMapView.getLongitudeSpan() / 4);
        mc.setCenter(pt);
    }

    public void panNorth() {
        GeoPoint pt = new GeoPoint(mMapView.getMapCenter().getLatitudeE6()
                + mMapView.getLatitudeSpan() / 4, mMapView.getMapCenter()
                .getLongitudeE6());
        mc.setCenter(pt);
    }

    public void panSouth() {
        GeoPoint pt = new GeoPoint(mMapView.getMapCenter().getLatitudeE6()
                - mMapView.getLatitudeSpan() / 4, mMapView.getMapCenter()
                .getLongitudeE6());
        mc.setCenter(pt);
    }

    public void zoomIn() {
        mc.zoomIn();
    }

    public void zoomOut() {
        mc.zoomOut();
    }

    public void toggleSatellite() {
        mMapView.setSatellite(true);
```

```
        mMapView.setStreetView(false);
        mMapView.setTraffic(false);
    }

    public void toggleTraffic() {
        mMapView.setTraffic(true);
        mMapView.setSatellite(false);
        mMapView.setStreetView(false);
    }

    public void toggleStreetView() {
        mMapView.setStreetView(true);
        mMapView.setSatellite(false);
        mMapView.setTraffic(false);
    }

    private void centerOnGPSPosition() {
        Log.d(TAG, "centerOnGPSPosition");
        String provider = "gps";
        LocationManager lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

        Location loc = lm.getLastKnownLocation(provider);
        loc = lm.getLastKnownLocation(provider);

        mDefPoint = new GeoPoint((int) (loc.getLatitude() * 1000000),
                (int) (loc.getLongitude() * 1000000));
        mDefCaption = "I'm Here.";
        mc.animateTo(mDefPoint);
        mc.setCenter(mDefPoint);
        // show Overlay on map
        MyOverlay mo = new MyOverlay();
        mo.onTap(mDefPoint, mMapView);
        mMapView.getOverlays().add(mo);
    }

    // This is used draw an overlay on the map
    protected class MyOverlay extends Overlay {
        @Override
        public void draw(Canvas canvas, MapView mv, boolean shadow) {
            Log.d(TAG, "MyOverlay::darw..mDefCaption=" + mDefCaption);
            super.draw(canvas, mv, shadow);

            if (mDefCaption.length() == 0) {
                return;
            }

            Paint p = new Paint();
            int[] scoords = new int[2];
            int sz = 5;

            // Convert to screen coords
```

```
            Point myScreenCoords = new Point();
            mMapView.getProjection().toPixels(mDefPoint, myScreenCoords);
            // mMapView.set
            // mv.getPointXY(mDefPoint, scoords);
            // Draw point caption and its bounding rectangle
            scoords[0] = myScreenCoords.x;
            scoords[1] = myScreenCoords.y;
            p.setTextSize(14);
            p.setAntiAlias(true);

            int sw = (int) (p.measureText(mDefCaption) + 0.5f);
            int sh = 25;
            int sx = scoords[0] - sw / 2 - 5;
            int sy = scoords[1] - sh - sz - 2;
            RectF rec = new RectF(sx, sy, sx + sw + 10, sy + sh);

            p.setStyle(Style.FILL);
            p.setARGB(128, 255, 0, 0);

            canvas.drawRoundRect(rec, 5, 5, p);

            p.setStyle(Style.STROKE);
            p.setARGB(255, 255, 255, 255);
            canvas.drawRoundRect(rec, 5, 5, p);
            // canvas.d

            canvas.drawText(mDefCaption, sx + 5, sy + sh - 8, p);

            // Draw point body and outer ring
            p.setStyle(Style.FILL);
            p.setARGB(88, 255, 0, 0);
            p.setStrokeWidth(1);
            RectF spot = new RectF(scoords[0] - sz, scoords[1] + sz, scoords[0]
                    + sz, scoords[1] - sz);
            canvas.drawOval(spot, p);

            p.setARGB(255, 255, 0, 0);
            p.setStyle(Style.STROKE);
            canvas.drawCircle(scoords[0], scoords[1], sz, p);
        }
    }

    //
    protected class MyLocationListener implements LocationListener {

        @Override
        public void onLocationChanged(Location loc) {
            Log.d(TAG, "MyLocationListener::onLocationChanged...");
            if (loc != null) {
                Toast.makeText(
                        getBaseContext(),
```

```
                        "Location changed : Lat: " + loc.getLatitude()
                                + " Lng: " + loc.getLongitude(),
                        Toast.LENGTH_SHORT).show();
                // Set up the overlay controller
                // mOverlayController = mMapView.createOverlayController();
                mDefPoint = new GeoPoint((int) (loc.getLatitude() * 1000000),
                        (int) (loc.getLongitude() * 1000000));
                mc.animateTo(mDefPoint);
                mc.setCenter(mDefPoint);
                // show on the map
                mDefCaption = "Lat: " + loc.getLatitude() + ",Lng: "
                        + loc.getLongitude();
                MyOverlay mo = new MyOverlay();
                mo.onTap(mDefPoint, mMapView);
                mMapView.getOverlays().add(mo);
                // //////////
                //if(mlcDbHelper == null){
                //      mlcDbHelper.open();
                //}
                //mlcDbHelper.createLocate(track_id,   loc.getLongitude(),loc.getLatitude(), loc.getAltitude());
            }
        }

        @Override
        public void onProviderDisabled(String provider) {
            Toast.makeText(
                    getBaseContext(),
                    "ProviderDisabled.",
                    Toast.LENGTH_SHORT).show();         }

        @Override
        public void onProviderEnabled(String provider) {
            Toast.makeText(
                    getBaseContext(),
                    "ProviderEnabled,provider:"+provider,
                    Toast.LENGTH_SHORT).show();         }

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {
            // TODO Auto-generated method stub
        }
    }

    //初始化菜单
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        menu.add(0, MENU_CON, 0, R.string.menu_con).setIcon(
                R.drawable.con_track).setAlphabeticShortcut('C');
        menu.add(0, MENU_DEL, 0, R.string.menu_del).setIcon(R.drawable.delete)
                .setAlphabeticShortcut('D');
```

```
        menu.add(0, MENU_NEW, 0, R.string.menu_new).setIcon(
                R.drawable.new_track).setAlphabeticShortcut('N');
        menu.add(0, MENU_MAIN, 0, R.string.menu_main).setIcon(R.drawable.icon)
                .setAlphabeticShortcut('M');
        return true;
    }

    //当一个菜单被选中时调用
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        Intent intent = new Intent();
        switch (item.getItemId()) {
        case MENU_NEW:
            intent.setClass(ShowTrack.this, NewTrack.class);
            startActivity(intent);
            return true;
        case MENU_CON:
            // TODO: 继续跟踪选择的记录
            startTrackService();
            return true;
        case MENU_DEL:
             mDbHelper = new TrackDbAdapter(this);
             mDbHelper.open();
            if (mDbHelper.deleteTrack(rowId)) {
                mDbHelper.close();
                intent.setClass(ShowTrack.this, iTracks.class);
                startActivity(intent);
            }else{
                mDbHelper.close();
            }
            return true;
        case MENU_MAIN:
            intent.setClass(ShowTrack.this, iTracks.class);
            startActivity(intent);
            break;
        }
        return true;
    }

    @Override
    protected void onStop() {
        super.onStop();
        Log.d(TAG, "onStop");
        //mDbHelper.close();
        //mIcDbHelper.close();
    }

    @Override
   public void onDestroy() {
        Log.d(TAG, "onDestroy.");
       super.onDestroy();
```

```
            stopTrackService();
        }
}
```

上述代码的具体实现流程如下。

第 1 步：通过 findViews 来确定要使用的控件，并绑定需要响应的事件。

第 2 步：通过 findViews 实现对地图的处理，首先获取布局中的 MapView，使用 getController 得到一个 MapController，然后注册一个基于 locationListener 的 MyLocationListener。

第 3 步：实现按钮的具体处理方法的代码，实现原理比较简单，即首先获取地图中心，然后向 4 个方向移动 1/4 距离。

第 4 步：通过单击 GPS 按钮的响应方法 centerOnGPSPosition()，将地图定位到当前 GPS 指定的位置。

第 5 步：通过 Overlay 抽象类重载实现其 draw()方法。

至此，本模块主要代码介绍完毕，执行后的效果如图 13-10 所示。

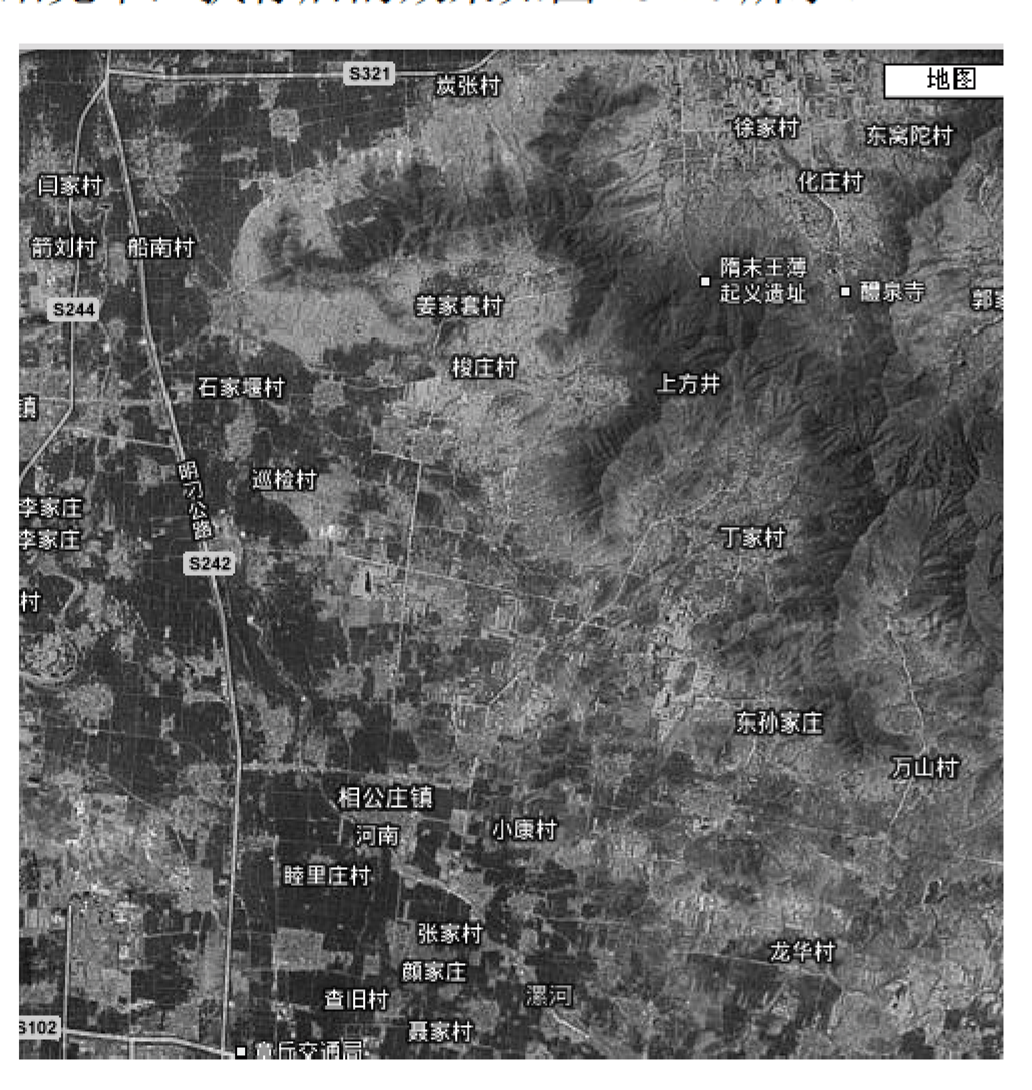


图 13-10　地图展示界面

13.2.7　数据存取

在前面介绍的系统需求分析中，系统要求将每次跟踪的目标位置保存在数据库中，并且每次改变后都要保存起来。本项目的个性化配置信息保存在 SharedPreferences 中，在此将需要存取的数据放在数据库中。在 Android 中，存取数据库的方法有两种，一种通过 help 类继承 SQLiteDatabase 相关类绑定 SQL，另外一种是使用 ContentProvider 进行封装。

1. 创建数据库

本项目需要同时操作数据库中的两个表，在此先在文件 DbAdapter.java 中创建一个名为 DbAdapter 的类，具体实现代码如下所示。

```
package com.iceskysl.map;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
```

```
import android.util.Log;

public class DbAdapter {
	private static final String TAG = "DbAdapter";
	private static final String DATABASE_NAME = "iTracks.db";
	private static final int DATABASE_VERSION = 1;

	public class DatabaseHelper extends SQLiteOpenHelper {
		public DatabaseHelper(Context context) {
			super(context, DATABASE_NAME, null, DATABASE_VERSION);
		}
		@Override
		public void onCreate(SQLiteDatabase db) {
			String tracks_sql = "CREATE TABLE " + TrackDbAdapter.TABLE_NAME + " ("
			+ TrackDbAdapter.ID + " INTEGER primary key autoincrement, "
			+ TrackDbAdapter.NAME + " text not null, "
			+ TrackDbAdapter.DESC + " text ,"
			+ TrackDbAdapter.DIST + " LONG ,"
			+ TrackDbAdapter.TRACKEDTIME + " LONG ,"
			+ TrackDbAdapter.LOCATE_COUNT + " INTEGER, "
			+ TrackDbAdapter.CREATED + " text, "
			+ TrackDbAdapter.AVGSPEED + " LONG, "
			+ TrackDbAdapter.MAXSPEED + " LONG ,"
			+ TrackDbAdapter.UPDATED + " text "
			+ ");";
			Log.i(TAG, tracks_sql);
			db.execSQL(tracks_sql);

			String locats_sql = "CREATE TABLE " + LocateDbAdapter.TABLE_NAME + " ("
			+ LocateDbAdapter.ID + " INTEGER primary key autoincrement, "
			+ LocateDbAdapter.TRACKID   + " INTEGER not null, "
			+ LocateDbAdapter.LON + " DOUBLE ,"
			+ LocateDbAdapter.LAT + " DOUBLE ,"
			+ LocateDbAdapter.ALT + " DOUBLE ,"
			+ LocateDbAdapter.CREATED + " text "
			+ ");";
			Log.i(TAG, locats_sql);
			db.execSQL(locats_sql);
		}
		@Override
		public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
			db.execSQL("DROP TABLE IF EXISTS " + LocateDbAdapter.TABLE_NAME + ";");
			db.execSQL("DROP TABLE IF EXISTS " + TrackDbAdapter.TABLE_NAME + ";");
			onCreate(db);
		}
	}

}
```

在上述代码中，重新定义了 SQLiteOpenHelper 的 onCreate()方法和 onUpgrade()方法，通过这两个方法实现了创建和升级数据库的脚本。

2. 数据库操作

本功能实现对两个表操作的封装处理，因为共用了同一个数据库，所以只需从前面创建的 DbAdapter 中继续继承即可，在此继承了两个类：TrackDbAdapter 和 LocateDbAdapter。通过对这两个类的封装，实现对数据表的操作。

（1）TrackDbAdapter 类是在文件 TrackDbAdapter.java 中定义的，主要代码如下所示。

```
package com.iceskysl.map;

import java.util.Calendar;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;

public class TrackDbAdapter extends DbAdapter{
	private static final String TAG = "TrackDbAdapter";

	public static final String TABLE_NAME = "tracks";

	public static final String ID = "_id";
	public static final String KEY_ROWID = "_id";
	public static final String NAME = "name";
	public static final String DESC = "desc";
	public static final String DIST = "distance";
	public static final String TRACKEDTIME = "tracked_time";
	public static final String LOCATE_COUNT = "locats_count";
	public static final String CREATED = "created_at";
	public static final String UPDATED = "updated_at";
	public static final String AVGSPEED = "avg_speed";
	public static final String MAXSPEED = "max_speed";

	private DatabaseHelper mDbHelper;
	private SQLiteDatabase mDb;
	private final Context mCtx;

	public TrackDbAdapter(Context ctx) {
		this.mCtx = ctx;
	}

	public TrackDbAdapter open() throws SQLException {
		mDbHelper = new DatabaseHelper(mCtx);
		mDb = mDbHelper.getWritableDatabase();
		return this;
	}

	public void close() {
		mDbHelper.close();
```

```
    }

    public Cursor getTrack(long rowId) throws SQLException {
        Cursor mCursor =
        mDb.query(true, TABLE_NAME, new String[ ] { KEY_ROWID, NAME,
                    DESC, CREATED }, KEY_ROWID + "=" + rowId, null, null,
                    null, null, null);
        if (mCursor != null) {
            mCursor.moveToFirst();
        }
        return mCursor;
    }

    public long createTrack(String name, String desc) {
        Log.d(TAG, "createTrack.");
        ContentValues initialValues = new ContentValues();
        initialValues.put(NAME, name);
        initialValues.put(DESC, desc);
        Calendar calendar = Calendar.getInstance();
        String created = calendar.get(Calendar.YEAR) + "-" +calendar.get(Calendar.MONTH) + "-" +
calendar.get(Calendar.DAY_OF_MONTH) + " "
                    + calendar.get(Calendar.HOUR_OF_DAY) + ":"
                    + calendar.get(Calendar.MINUTE) + ":"   + calendar.get(Calendar.SECOND);
        initialValues.put(CREATED, created);
        initialValues.put(UPDATED, created);
        return mDb.insert(TABLE_NAME, null, initialValues);
    }

    //
    public boolean deleteTrack(long rowId) {
        return mDb.delete(TABLE_NAME, KEY_ROWID + "=" + rowId, null) > 0;
    }

    public Cursor getAllTracks() {
        return mDb.query(TABLE_NAME, new String[ ] { ID, NAME,
                    DESC, CREATED }, null, null, null, null, "updated_at desc");
    }

    public boolean updateTrack(long rowId, String name, String desc) {
        ContentValues args = new ContentValues();
        args.put(NAME, name);
        args.put(DESC, desc);
        Calendar calendar = Calendar.getInstance();
        String updated = calendar.get(Calendar.YEAR) + "-" +calendar.get(Calendar.MONTH) + "-" +
calendar.get(Calendar.DAY_OF_MONTH) + " "
        + calendar.get(Calendar.HOUR_OF_DAY) + ":"
        + calendar.get(Calendar.MINUTE) + ":"   + calendar.get(Calendar.SECOND);
        args.put(UPDATED, updated);
        return mDb.update(TABLE_NAME, args, KEY_ROWID + "=" + rowId, null) > 0;
    }

}
```

在上述代码中，首先声明了一些常量，然后根据需要的操作功能定义了具体方法。

（2）LocateDbAdapter 类是在文件 LocateDbAdapter.java 中实现的，主要代码如下所示。

```
package com.iceskysl.map;

import java.util.Calendar;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;

public class LocateDbAdapter extends DbAdapter {
	private static final String TAG = "LocateDbAdapter";
	public static final String TABLE_NAME = "locates";

	public static final String ID = "_id";
	public static final String TRACKID = "track_id";
	public static final String LON = "longitude";
	public static final String LAT = "latitude";
	public static final String ALT = "altitude";
	public static final String CREATED = "created_at";

	private DatabaseHelper mDbHelper;
	private SQLiteDatabase mDb;
	private final Context mCtx;

	public LocateDbAdapter(Context ctx) {
		this.mCtx = ctx;
	}

	public LocateDbAdapter open() throws SQLException {
		mDbHelper = new DatabaseHelper(mCtx);
		mDb = mDbHelper.getWritableDatabase();
		return this;
	}

	public void close() {
		mDbHelper.close();
	}

	public Cursor getLocate(long rowId) throws SQLException {
		Cursor mCursor =
		mDb.query(true, TABLE_NAME, new String[] { ID, LON,
				LAT, ALT, CREATED }, ID + "=" + rowId, null, null,
				null, null, null);
		if (mCursor != null) {
			mCursor.moveToFirst();
		}
		return mCursor;
```

```
	}

	public long createLocate(int track_id, Double longitude, Double latitude ,Double altitude) {
		Log.d(TAG, "createLocate.");
		ContentValues initialValues = new ContentValues();
		initialValues.put(TRACKID, track_id);
		initialValues.put(LON, longitude);
		initialValues.put(LAT, latitude);
		initialValues.put(ALT, altitude);

		Calendar calendar = Calendar.getInstance();
		String created = calendar.get(Calendar.YEAR) + "-" +calendar.get(Calendar.MONTH) + "-" +
calendar.get(Calendar.DAY_OF_MONTH) + " "
				+ calendar.get(Calendar.HOUR_OF_DAY) + ":"
				+ calendar.get(Calendar.MINUTE) + ":"	+ calendar.get(Calendar.SECOND);
		initialValues.put(CREATED, created);
		return mDb.insert(TABLE_NAME, null, initialValues);
	}

	//
	public boolean deleteLocate(long rowId) {
		return mDb.delete(TABLE_NAME, ID + "=" + rowId, null) > 0;
	}


	public Cursor getTrackAllLocates(int trackId) {
		return mDb.query(TABLE_NAME, new String[ ] { ID,TRACKID, LON,
				LAT, ALT,CREATED }, "track_id=?", new String[ ] {String.valueOf(trackId)}, null, null,
"created_at asc");
	}
}
```

在上述代码中，也是首先声明了一些常量，然后根据需要的操作功能定义了具体方法。

13.2.8 实现 Service 服务

项目的基本要求是，切换一个界面不会影响到对目标的追踪，也就是说，即使来到另外一个界面，程序也需要在后台进行跟踪和记录，所以需要用到 Service 服务。首先在文件 AndroidManifest.xml 中加入对 Service 的声明，代码如下。

```
<service android:name=".Track">
	<intent-filter>
		<action android:name="com.iceskysl.map.START_TRACK_SERVICE" />
		<category android:name="android.intent.category.default" />
	</intent-filter>
</service>
```

通过上述代码添加了一个名为 Track 的 Service，并设定了其名字为 com.iceskysl.map.START_TRACK_SERVICE。处理文件 Track.java 的具体实现代码如下。

```
package com.iceskysl.map;

import android.app.Service;
```

```
import android.content.Context;
import android.content.Intent;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.os.IBinder;
import android.util.Log;
import android.widget.Toast;

public class Track extends Service {
    private static final String TAG = "Track";

    private LocationManager lm;
    private LocationListener locationListener;

    static LocateDbAdapter mlcDbHelper = null;
    private int track_id;

    @Override
    public IBinder onBind(Intent arg0) {
        Log.d(TAG, "onBind.");
        return null;
    }

    public void onStart(Intent intent, int startId) {
        Log.d(TAG, "onStart.");
      super.onStart(intent, startId);
      startDb();
        Bundle extras = intent.getExtras();
        if (extras != null) {
            track_id = extras.getInt(LocateDbAdapter.TRACKID);
        }
        Log.d(TAG, "track_id =" + track_id);
        //use the LocationManager class to obtain GPS locations
        lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
        locationListener = new MyLocationListener();
        lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
                locationListener);
   }

    private void startDb() {
        if(mlcDbHelper == null){
            mlcDbHelper = new LocateDbAdapter(this);
            mlcDbHelper.open();
        }
    }

    private void stopDb() {
        if(mlcDbHelper != null){
           mlcDbHelper.close();
        }
    }
```

```
        public static LocateDbAdapter getDbHelp(){
            return mlcDbHelper;
        }


    public void onDestroy() {
            Log.d(TAG, "onDestroy.");
        super.onDestroy();
        stopDb();
    }
    protected class MyLocationListener implements LocationListener {

            @Override
            public void onLocationChanged(Location loc) {
                Log.d(TAG, "MyLocationListener::onLocationChanged...");
                if (loc != null) {
                    // //////////
                    if(mlcDbHelper == null){
                        mlcDbHelper.open();
                    }
                    mlcDbHelper.createLocate(track_id, loc.getLongitude(),loc.getLatitude(), loc.getAltitude());
                }
            }

            @Override
            public void onProviderDisabled(String provider) {
                Toast.makeText(
                        getBaseContext(),
                        "ProviderDisabled.",
                        Toast.LENGTH_SHORT).show();          }

            @Override
            public void onProviderEnabled(String provider) {
                Toast.makeText(
                        getBaseContext(),
                        "ProviderEnabled,provider:"+provider,
                        Toast.LENGTH_SHORT).show();          }
            @Override
            public void onStatusChanged(String provider, int status, Bundle extras) {
                // TODO Auto-generated method stub
            }
        }
}
```

在上述代码中，Track 继承了 Service 类，然后在 onStart()中连接数据库，接收了参数并设定监听器，使用 MyLocationListener，当位置变化（onLoacationChanged）时，调用前面数据存储部分已经实现的 mlcDbHelper.createLocate()方法，将位置信息和接收到的参数写入到数据库中。

第 14 章　QQ 聊天记录查看器

1999 年 2 月，腾讯公司正式推出第一款即时通信软件——OICQ，后改名为腾讯 QQ。腾讯 QQ 支持在线聊天、视频聊天、语音聊天、点对点断点续传文件、共享文件、自定义面板、远程控制、QQ 邮箱、传送离线文件等多种功能，并可与移动通信终端等多种通信方式相连。本章将详细讲解在 Android 系统中 QQ 漏洞的问题，并通过具体实例讲解开发 QQ 聊天记录查看器的具体过程。

14.1　Android 安全机制概述

知识点讲解：光盘：视频\视频讲解\第 14 章\Android 安全机制概述.avi

根据 Android 系统架构分析，其安全机制是基于 Linux 操作系统的内核安全机制基础上的，这主要体现在如下两个方面。

（1）使用进程沙箱机制来隔离进程资源。

（2）通过 Android 系统独有的内存管理技术，安全高效地实现进程之间的通信处理。

上述安全机制策略十分适合嵌入式移动终端处理器设备，因为这可以很好地兼顾高性能与内存容量的限制。另外，因为 Android 应用程序是基于 Framework 应用框架的，并且使用 Java 语言进行编写，所以最后运行于 Dalvik VM（Android 虚拟机）。同时，Android 的底层应用由 C/C++语言实现，以原生库形式直接运行于操作系统的用户空间。这样，Android 应用程序和 Dalvik VM 的运行环境都被控制在“进程沙箱”环境下。“进程沙箱”是一个完全被隔离的环境，拥有专用的文件系统区域，能够独立共享私有数据。Android 安全机制架构如图 14-1 所示。

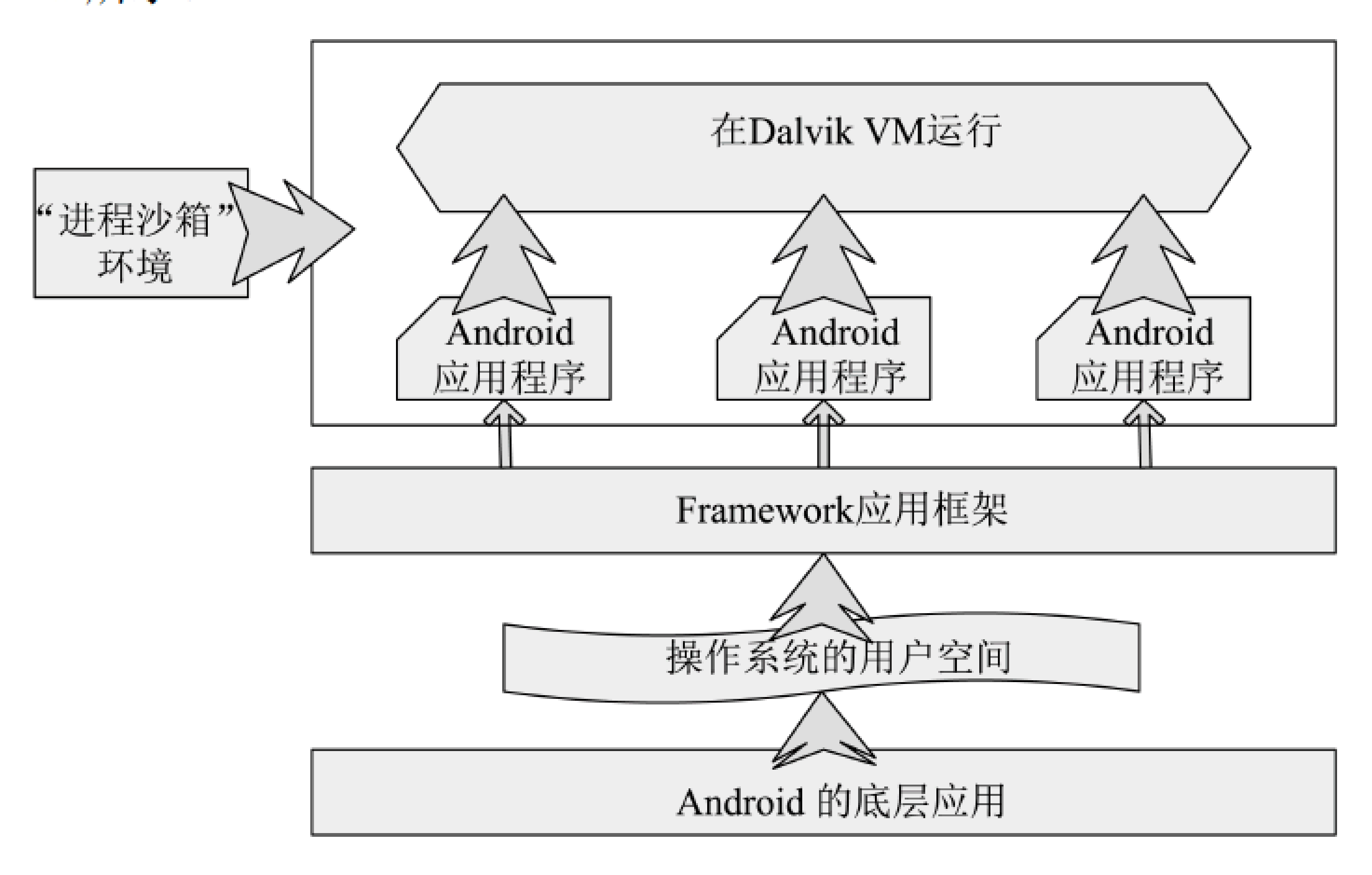


图 14-1　Android 安全机制架构

本节将详细讲解 Android 安全机制的基本架构知识。

14.1.1 Android 的安全机制模型

在 Android 系统的应用层中，提供了如下安全机制模型。

☑ 使用显式定义经用户授权的应用权限控制机制的方法，系统规范并强制各类应用程序的行为准则与权限许可。

☑ 提供应用程序的签名机制，实现应用程序之间的信息信任和资源共享。

概览整个 Android 系统的框架结构，其安全机制的具体特点如下。

☑ 采用不同的层次架构机制来保护用户信息的安全，并且不同的层次可以保证各种应用的灵活性。

☑ 鼓励更多的用户去了解应用程序的工作过程，鼓励用户花费更多的时间和注意力来关注移动设备的安全性。

☑ 无惧恶意软件的威胁，并拥有坚定的信念消除这些威胁。

☑ 时刻防范第三方恶意应用程序的攻击。

☑ 时刻做好风险控制工作，一旦安全防护系统崩溃，要尽量减少损失，并尽快恢复。

根据上述模型，Android 安全系统提供了如下安全机制。

（1）内存管理

Android 内存管理机制基于标准 Linux 的 OOM（低内存管理）机制，实现低内存清理（LMK）机制，将所有的进程按照重要性进行分级，系统会自动清理最低级别进程所占用的内存空间。另外，还引用 Android 独有的共享内存机制 Ashmem，此机制具有清理不再使用共享内存区域的能力。

（2）权限声明

Android 应用程序需要显式声明权限、名称、权限组与保护级别，只有这样才能算是一个合格的 Android 程序。在 Android 系统中规定：不同级别应用程序的使用权限的认证方式不同，具体说明如下。

☑ Normal 级：申请后即可用。

☑ Dangerous 级：在安装时由用户确认后方可用。

☑ Signature 与 Signatureorsystem 级：必须是系统用户才可用。

（3）应用程序签名

Android 应用程序包（.apk 格式文件）必须被开发者数字签名，同一名开发者可以指定不同的应用程序共享 UID，这样可以运行在同一个进程空间以实现资源共享。

（4）访问控制

通过使用基于 Linux 系统的访问控制机制，可以确保系统文件与用户数据不被非法访问。

（5）进程沙箱隔离

当安装 Android 应用程序时会被赋予一个独特的用户标识（UID），这个标识永久保持。当 Android 应用程序及其运行的 Dalvik VM 运行于独立的 Linux 进程空间时，会将与 UID 不同的应用程序隔离出来。

（6）进程通信

Android 采用 Binder 机制提供的共享内存实现进程通信功能，Binder 机制基于 Client-Server 模式，提供类似于 COM 和 CORBA 的轻量级远程进程调用（RPC）。通过使用 Binder 机制中的接口描述语言（AIDL）来定义接口与交换数据的类型，这样可以确保进程间通信的数据不会发生越界操作，影响进程的空间。

14.1.2 Android 具有的权限

Android 安全结构的核心为“应用程序在默认的情况下不可以执行任何对其他应用程序、系统或者用户带来负面影响的操作。”作为开发者来说，只有了解并把握 Android 安全架构的核心，才能有在使用过程中

用户体验更加流畅的设计。

根据用户的使用过程体验，可以将和 Android 系统相关的权限分为如下 3 类。

☑ Android 手机所有者权限：自用户购买 Android 手机（如 Samsung GT-i9000）后，用户不需要输入任何密码，就具有安装一般应用软件、使用应用程序等的权限。

☑ Android root 权限：该权限为 Android 系统的最高权限，可以对系统中所有文件、数据进行任意操作。出厂时默认没有该权限，需要使用 z4Root 等软件获取，然而，并不鼓励进行此操作，因为可能因此使用户失去手机原厂保修的权益。同样，如果将 Android 手机进行 root 权限提升，则此后用户不需要输入任何密码，都能以 Android root 权限来使用手机。

☑ Android 应用程序权限：Android 提供了丰富的 SDK（Software Development Kit），开发人员可以根据其开发 Android 中的应用程序。而应用程序对 Android 系统资源的访问需要有相应的访问权限，这个权限就称为 Android 应用程序权限，在应用程序设计时设定，在 Android 系统中初次安装时即生效。值得注意的是，如果应用程序设计的权限大于 Android 手机所有者权限，则该应用程序无法运行。例如，没有获取 Android root 权限的手机无法运行 Root Explorer，因为运行该应用程序需要 Android root 权限。

14.1.3　Android 的组件模型（Component Model）

整个 Android 系统中包括 4 种组件，具体说明如下。

☑ Activity：是一个界面，这个界面中可以放置各种控件。例如，Task Manager 的界面、Root Explorer 的界面等。

☑ Service：服务是运行在后台的功能模块，如文件下载、音乐播放程序等。

☑ Content Provider：是 Android 平台应用程序间数据共享的一种标准接口，以类似于 URI（Universal Resources Identification）的方式来表示数据，例如，content://contacts/people/1101。

☑ Broadcast Receiver：与 Broadcast Receiver 组件相关的概念是 Intent，Intent 是一个对动作和行为的抽象描述，负责组件之间、程序之间进行消息传递。而 Broadcast Receiver 组件则提供了一种把 Intent 作为一个消息广播出去，由所有对其感兴趣的程序对其作出反应的机制。

14.1.4　Android 安全访问设置

在 Android 系统中，每个应用程序的 APK（Android Package）包中都包含一个 AndroidMainifest.xml 文件，该文件除了罗列应用程序运行时库、运行依赖关系之外，还会详细地罗列出该应用程序所需的系统访问。AndroidMainifest.xml 文件的基本格式如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="cn.com.fetion.android"
      android:versionCode="1"
      android:versionName="1.0.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".welcomActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

```
            </activity>
        </application>
        <uses-permission android:name="android.permission.SEND_SMS"></uses-permission>
    </manifest>
```

在上述代码中的加粗斜体部分，是声明该软件具备发送短信的功能。在 Android 系统中，一共定义了 100 多种 Permission（权限）供开发人员使用。

14.1.5　Linux 系统的安全机制

因为 Android 系统是基于 Linux 内核的，所以在学习 Android 的安全机制之前需要先掌握 Linux 安全机制的知识。本节将详细讲解 Linux 用户权限、进程和内存空间方面的知识，为读者学习本书后面的知识打下基础。

1. root 用户、伪装用户和普通用户

在 Linux 系统的安全机制中，最基础的是用户与用户组。Linux 系统的用户由用户名和用户标识（UID）表示，用户可同时参与多个用户组，每个用户组由组标识（GID）表示。在 Linux 系统中存在 3 类用户，分别是 root 用户、系统伪装用户和普通用户。

（1）root 用户

在 Linux 操作系统中的最高权限是 root，也称为超级权限的拥有者，此类用户具有最高的系统权限，UID 为 0。因为 root 用户能完成普通用户无法执行的操作，所以也将 root 用户称为超级管理用户。在 Linux 系统中，每个文件、目录和进程都归属于某一个用户，只有这个用户才能操作该文件、目录和进程。root 用户可以超越任何用户和用户组来对文件或目录进行读取、修改或删除（在系统正常的许可范围内），可以控制程序的执行和终止，可以对硬件设备进行添加、创建和移除等操作，也可以对文件和目录的属性和权限进行修改，以适合系统管理的需要（因为 root 是系统中权限最高的特权用户）。

在 Linux 系统中是通过 UID 来区分用户权限级别的，UID 为 0 的用户被系统约定为具有超级权限，也就是 root 用户。超级用户具有在系统约定的最高权限内操作的功能，所以说超级用户可以完成系统管理的所有工作；可以通过/etc/passwd 来查得 UID 为 0 的用户是 root，而且只有 root 对应的 UID 为 0，从这一点来看，root 用户在系统中有无可替代的至高地位和无限制权限。

当系统默认安装时，系统用户和 UID 是一对一关系，即一个 UID 对应一个用户。Linux 系统的用户身份是通过 UID 来确认的，UID 是确认用户权限的标识，用户登录系统所处的角色是通过 UID 来实现的，并不是用户名。如果几个用户共同使用同一个 UID ，则是一件很危险的事情。例如，将普通用户的 UID 改为 0，就造成了系统管理权限的混乱。如果想用 root 权限，可以通过 su 或 sudo 的方式来实现，而不是随意让一个用户和 root 来共享同一个 UID。

在 Linux 系统中，可以让 UID 和用户实现一对多的关系。例如，可以将一个 UID 为 0 的值分配给几个用户共同使用，这就实现了 UID 和用户的一对多的关系。但这样做会使相同 UID 的用户具有相同的身份和权限，会带来一定的风险。例如，在系统中把 guan 这个普通用户的 UID 改为 0 后，表示普通用户 guan 具有了超级权限，其权限能力和 root 用户完全一样。由此可见，UID 为 0 的用户就是 root ，root 用户的 UID 就是 0。

UID 和用户一对一的对应关系，是管理员进行系统管理时应坚守的准则之一，超级权限保留给 root 是唯一的也是最好的选择。如果不把 UID 的 0 值分享给其他用户使用，只有 root 用户拥有 UID=0，那么 root 用户就是唯一的超级权限用户。

（2）普通用户和伪装用户

与超级用户相对的就是普通用户和伪装用户（也称为虚拟用户），普通用户和伪装用户都是功能受限



的用户。为了完成特定的任务，Linux 系统必须提供普通用户和伪装用户。Linux 系统是一个多用户、多任务的操作系统，多用户主要体现在用户的角色的多样性，不同的用户所分配的权限也不同。其实这也是 Linux 系统比 Windows 系统更为安全的本质所在。

Linux 操作系统出于系统管理的需要，将某些关键系统应用文件所有权赋予某些系统伪装用户，其 UID 范围为 1～499，Linux 系统的伪装用户不能登录系统。

Linux 操作系统中的普通用户只具备有限的访问权限，UID 为 500～6000，可以登录系统获得 shell。

2. 超级用户（权限）

在 Linux 系统中，超级权限用户（UID 为 0 的用户）的主要功能如下。

（1）对任何文件、目录或进程进行操作

这种操作是在 Linux 系统最高许可范围内的操作，而有些操作即使具有超级权限的 root 也无法完成，例如/proc 目录，/proc 是用来反映系统运行的实时状态信息的，所以即使是 root 用户也无能为力，其权限如下所示。

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# cd /
[root@localhost /]# ls -ld /proc/
dr-xr-xr-x 134 root root 0 2005-10-27 /proc/
```

对于/proc 目录来说，root 用户只具有读和执行权限，但没有写权限。

（2）对于涉及系统全局的系统管理

在 Linux 系统中，对硬件管理、文件系统管理、用户管理和系统全局配置等操作都需要超级权限来实现，例如，使用 adduser 来添加新用户的功能就需通过超级权限的用户来完成。

（3）超级权限的不可替代性

因为超级权限在系统管理中是不可缺少的，所以必须使用超级权限来完成系统管理工作。在一般情况下，为了系统安全，不需要使用 root 用户来完成一般的应用。root 用户只是用来管理和维护系统功能，例如，系统日志的查看、清理，用户添加/删除等操作。在不涉及系统管理的情况下，普通用户足以完成基本功能，例如编写文件、收听音乐等。对于基于普通应用程序的调用工作，可以通过普通用户来完成。

当以普通权限的用户登录系统时，有些系统配置及系统管理必须通过超级权限用户完成，例如，对系统日志的管理、添加和删除用户。获取超级权限的过程，就是切换普通用户身份到超级用户身份的过程，这个过程主要通过 su 和 sudo 来解决。

3. 文件权限

Linux 系统中的用户对系统资源拥有具体的访问权限，例如文件资源。在 Linux 系统中，系统资源通常以“文件”来表示。Linux 中的“文件”不仅限于通常意义上存储于物理介质上的数据文件，还可以是已被抽象成用户程序访问系统资源的统一接口。通过对文件实现打开、关闭、读、写以及控制等操作，用户程序不但可以访问操作系统控制的各类设备，而且可以访问内核的数据资源与运行状态。例如，/dev 目录下的文件通常为系统硬件设备的访问接口，而/proc 下的文件通常是内核的进程控制信息访问接口。为了控制不同的用户对不同系统资源的访问，在 Linux 操作系统中使用不同的用户权限来实现访问控制。

（1）3 个组

在 Linux 权限机制下，每个文件属于一个用户和一个组，由 UID 与 GID 标识其具体的所有权。对于文件的具体访问权限，分别定义为可读（r）、可写（w）与可执行（x）3 大类，并且由 3 组读、写、执行组成的权限三元组来描述相关权限，具体说明如下。

☑ 第 1 组：定义文件所有者（用户）的权限。
☑ 第 2 组：定义同组用户（GID 相同但 UID 不同的用户）的权限。
☑ 第 3 组：定义其他用户的权限（GID 与 UID 都不同的用户），例如，通过如下命令可以获得文件的权限设置。

```
$ ls –l /bin/foobar
-rwxr-xr--    1 root wheel 20540 Oct 26 07:49 /bin/mmm
```

在上述命令中，“-rwxr-xr--”中的首字符“-”表示文件/bin/mmm 的类型，也有可能是如下表示不同文件类型的字符。

☑ d：目录。
☑ l：符号链接。
☑ c：字符设备文件。
☑ b：块设备文件。
☑ p：pipe 管道。
☑ s：socket 套接字。

字段“-rwxr-xr--”首字符后面是 3 个三元组，即 rwx 的组合，其中，“-”表示无相应权限，具体说明如下。

☑ rwx：表示文件所有者（此处为 root）可以进行读、写、执行的操作。
☑ r-x：表示文件组用户（此处为 wheel 组的用户）没有写权限，但有可读和执行权限。
☑ r--：表示任何其他用户对本文件的权限，即只可读，没有可写与执行的权限。

（2）特殊权限标识

Linux 安全机制对可执行的文件还提供了特殊的权限标识，分别是 suid、sgid 和“粘滞位”（sticky bit）。这 3 个元素实际成为上述权限三元组之外的第 4 个组，以 suid、sgid 和 stickybit 方式存在，具体说明如下。

☑ s 或 S（suid）：可执行文件启用此权限后可以得到获得该文件所有者的全部权限的特权，以该文件所有者的身份访问其能访问的全部资源。如果 suid 权限的文件被黑客利用，例如，以 suid 配上 root 拥有者，系统安全性将被破坏。
☑ s 或 S（sgid）：如果可执行文件启用此权限，则效果会与 suid 相同，即将文件所有者换成用户组，该文件就可以任意存取整个用户组所能使用的系统资源。
☑ t 或 T（stickybit）：在/tmp 和/var/tmp 目录中提供了供所有用户暂时的存取文件，用户都可以拥有完整的权限进入该目录，以便浏览、删除和移动自己的文件。

注意： 因为特殊权限会导致“特权”发生，所以说如果无特殊需求，就不应该启用这些权限，避免出现安全漏洞。

在 Linux 系统中，suid、sgid 和 stickybit 会占用 x 的位置，大小写有区分。如果开启执行权限（x），并且同时启用 suid、sgid、stickybit 中任意一个，则 s 采用小写替代 x。如果没有开启执行权限，仅启用 suid、sgid、stickybit 中任意一个，则 s 采用大写形式。例如，想查看修改用户口令的命令 passwd 的权限，可以通过如下指令实现。

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root wheel 17588    Oct 29 07:53    /usr/bin/passwd
```

在上述命令中，“-rwsr-xr-x”中的 s 替代了表示用户权限的第 1 个三元组中的 x，s 表示/bin/passwd 文件，被设置了 suid 和可执行位。如果 s 为大写，则表示文件只被设置了 suid。当运行 passwd 时，passwd 代表 root 用户执行，所以具有了超级用户访问权，而不再代表运行它的用户。运行/bin/passwd 文件时会更改/etc/passwd 文件的内容。尽管/etc/passwd 文件的所有者是 root，但是 suid 使得/bin/passwd 以 root 用户的访

问权限 lauren 运行，所以能够修改/etc/passwd 文件。同理，sgid 允许用户程序继承程序的组所有权，而不是当前用户的程序所有权。当 sgid 用于定义目录权限时，便启用了目录的 sgid 标志，在目录内创建的任何文件将继承目录的组。而“粘滞位”则通常用于目录的属性定义，现实中常用的/tmp 或/vat/tmp 目录等通常设置了“粘滞位”，例如，t 表示该目录设置了粘滞位。在为权限为 777 的目录/tmp 设置粘滞位后，具有写权限的每个用户都可以在目录下创建文件，不同的是每个用户只能删除自己是所有者的文件，即只能删除自己创建的文件。

例如，在如下指令中，如果/foo/mm1 已设置执行权限，则显示 s 与 t；否则/foo/mm2 没有执行权限，采用大写 S 和 T 来表示。

```
$ ls -ld /tmp
drwxrwxrwt 5 root root 4096 Oct 21 07:55 /tmp
$ ls -ld /foo/mm1
-rwsr-sr-t 1 root root 4096 Oct 21 07：17 /foo/mm1
$ ls -ld /foo/mm2
-rwSr-Sr-T 1 root root 4096 Oct 21 07：18 /foo/mm2
```

4. 使用 su 命令临时切换用户身份

在 Linux 系统中，su 命令就是切换用户的工具。例如，当以普通用户 guan 登录系统后，如果要添加用户任务以执行 useradd，guan 用户是没有这个权限的，而这个权限恰恰由 root 所拥有。解决上述问题的办法有两个，具体说明如下。

（1）退出 guan 用户，重新以 root 用户登录。

（2）不退出 guan 用户，而是用 su 切换到 root 下进行添加用户的工作，等任务完成后再退出 root。

在上述两种实现方法中，可以发现通过 su 切换的方式比较简单易行。

在 Linux 系统中，通过 su 可以在用户之间实现切换操作。当超级权限用户 root 向普通或虚拟用户切换时不需要密码，而当普通用户切换到其他任何用户时都需要密码验证。在 Linux 系统中，使用 su 的语法格式如下所示。

```
su [OPTION 选项参数] [用户]
```

☑ ??-, -l, --login：登录并改变到所切换的用户环境。

☑ ??-c, --commmand=COMMAND：执行一个命令，然后退出所切换到的用户环境。

在 Linux 系统中，su 在不加任何参数时表示默认切换到 root 用户，但是并没有被转到 root 用户目录下。也就是说，这时虽然切换为 root 用户，但是并没有改变 root 的登录环境。可以在/etc/passwd 中得到用户默认的登录环境，包括目录和 SHELL 定义等信息，例如：

```
[beinan@localhost ~]$ su +
Password:
[root@localhost beinan]# pwd
/home/guan
```

其中，su 加参数“-”表示默认切换到 root 用户，并且改变到 root 用户的环境，例如：

```
[beinan@localhost ~]$ pwd
/home/guan
[beinan@localhost ~]$ su -
Password:
[root@localhost ~]# pwd
/root
```

其中，su 参数“-”表示用户名，例如：

```
[beinan@localhost ~]$ su - root
Password:
```

```
[root@localhost ~]# pwd
/root
```

（3）su 的优缺点分析

在 Linux 系统中，su 为管理带来了方便，通过切换到 root 下的方式能够完成所有的系统管理。只要把 root 密码交给任何一个普通用户，就可以切换到 root 来完成所有的系统管理工作。但是通过 su 切换到 root 时也会存在不安全的因素，例如，系统有 10 个用户，而且都参与管理。如果这 10 个用户都涉及超级权限的运用，作为管理员，如果想让其他用户通过 su 来切换到超级权限的 root，必须把 root 权限密码都告诉这 10 个用户。如果这 10 个用户都拥有 root 权限，那么就都可以通过 root 权限做任何操作，这在一定程度上对系统的安全造成了威胁。无法保证这 10 个用户都能按照正常的操作流程来管理系统，任何一个用户对系统操作的重大失误都可能导致系统崩溃或数据丢失。

由此可见，在多人参与的系统管理中，su 工具并不是最好的选择，因此 su 只适用于一两个人参与管理的系统。

5. 进程

在 Linux 系统中，每个执行的任务都称为进程（Process），例如，使用 ls 命令浏览目录内容，或查询日期时间时输入的 date 命令。在每个进程启动时，系统都会为其指定一个唯一的数值，该数值就称为“进程 ID”（Process ID，PID）。如果要针对某个进程进行管理，例如结束进程的执行，必须以进程 ID（而不是该进程的名称）作为参考的对象。每个 Linux 进程都会存在一个对应的父进程（Parent Process），由这个父进程可以复制多个子进程，这是编写网络程序时很常用的一种方式，这个动作称为 Fork（孕育）。在现实应用中，最常见的一个 Fork 示例就是 Web 服务器，Web 服务器通常都可以支持多个客户端的连接，而服务器方面利用一个父进程来接受客户端的请求，然后利用 Fork 产生一个子进程以处理后续的任务，之后该父进程就可再度回到等待客户端请求的状态，如此即可不断地为客户端服务，如图 14-2 所示。

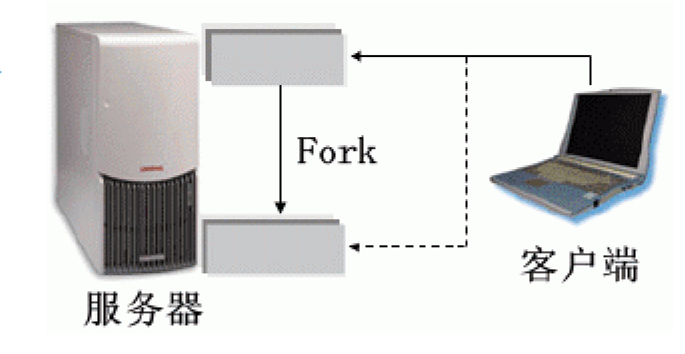


图 14-2　孕育过程

（1）前台与后台进程

在 Linux 系统中，每个进程都可能以两种方式存在：前台（Foreground）与后台（Background）。所谓前台进程，就是用户目前在屏幕上进行操作的进程；而后台进程则是实际上在操作，但在屏幕上无法看到的进程。通常使用后台方式执行的情况是，当此进程较为复杂且必须执行较长的时间时，会将其置于后台中执行，以避免占用屏幕的时间过久，而无法执行其他进程。

系统的服务一般都是以后台进程的方式存在的，而且都会驻留在系统中，直到关机时才结束，这类服务也称为 Daemon，在 Linux 系统中就包含许多 Daemon。判断 Daemon 最简单的方法就是由名称来判断，多数 Daemon 都是由服务名称加上 d 来产生的，例如，HTTP 服务的 Daemon 为 httpd。

（2）显示目前进程

在 Linux 系统中，ps 命令是 Process Status 的缩写，其功能是查看目前系统中有哪些进程正在执行，以及各进程的执行情况。可以直接输入 ps 命令名称，而无需在前面加任何参数。如果直接执行 ps 命令，则会发现类似如下所示的信息。

```
[root@ns1 ~]# ps
PID TTY          TIME CMD
1635 pts/0       00:00:00 su
1636 pts/0       00:00:00 bash
1679 pts/0       00:00:00 ps
```

上述 ps 命令的功能是显示 4 个字段的数据，具体说明如下。

☑ PID：进程标识（Process ID），系统是凭着这个编号来识别及处理此进程的。



☑ TTY：Teletypewriter，登录的终端机编号。
☑ TIME：此进程所消耗的 CPU 时间。
☑ CMD：正在执行的命令或进程名称。

上述的信息是 ps 命令显示的最基本数据情形，其实 ps 支持非常多的参数。

（3）显示详细信息

在 Linux 系统中，如果需显示更加详细的系统数据，可以使用-l（Long）参数实现。这样除了显示 ps 命令的 4 个基本字段数据外，还有 10 个额外数据可供查看，这些额外数据的内容及说明如下。

```
root@ns1 ~]# ps -l
F S   UID    PID   PPID C PRI NI ADDR   SZ WCHAN TTY TIME     CMD
4 S       0    9822    9521 0    81    0       -    1220 wait4    pts/2 00:00:00    su
4 S       0    9970    9822 0    75    0       -    1294 wait4    pts/2 00:00:00b ash
4 R       0 15354    9970 0    80    0       -      788  -          pts/2 00:00:00    ps
```

其中，F 表示该进程状态的标志（Flag），常用标志的具体说明如下。

☑ ALIGNWARN：标识代码是 001，表示打印警告信息。
☑ STARTING：标识代码是 002，表示进程正在初始化。
☑ EXITING：标识代码是 004，表示系统正在关机。
☑ PTRACED：标识代码是 010，表示已调用 ptrace（0）。
☑ TRACESYS：标识代码是 020，表示跟踪 System Call。
☑ FORKNOEXEC：标识代码是 040，表示已执行 fork 但没有执行 exec。
☑ SUPERPRIV：标识代码是 100，表示以 root 身份执行。
☑ DUMPCORE：标识代码是 200，表示内核转储。
☑ SIGNALED：标识代码是 400，表示以 Signal 结束进程。

而 S 表示进程状态代码（Process State Codes），可用的代码及说明如下。

☑ D：表示不可中断的闲置状态（Uninterruptible Sleep）。
☑ R：表示可执行的状态。
☑ S：表示闲置状态。
☑ T：表示跟踪或停止。
☑ Z：表示已死亡的进程（Zombie）。
☑ W：表示没有足够的内存页可分配。
☑ <：表示高优先级的进程。
☑ N：表示低优先级的进程。
☑ L：表示有内存页分配并锁在内存内。
☑ UID：进程执行者的 ID（User ID）。
☑ PPID：父进程标识（Parent Process ID）。
☑ PRI：表示进程执行的优先级（Priority）。
☑ NI：表示 Nice，是指进程执行优先级的 nice 值，负值表示其优先级较高。
☑ SZ：表示 Size，进程所占用的内存大小，以 KB 为单位。
☑ WCHAN：表示 Waiting Channel，是进程或系统调用等待时的地址。

另一种显示详细内容信息的参数为-u（User），其主要功能是以用户格式来显示进程数据，下面是部分示例内容以及新的字段说明。

```
[root@ns1 ~]# ps -u
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root    9822    0.0     0.0    4880 168 pts/2    S    12:20    0:00     [su]
```

```
root   9970   0.0    0.4   5176 872 pts/2   S   12:20   0:00    -bash
root 15448   0.0    0.3   2644 696 pts/2   R   12:30   0:00    ps -u
...
?%CPU：CPU 使用率百分比
?%MEM：内存使用率百分比
?VSZ：占用的虚拟内存大小
?RSS：占用的物理内存大小
?START：进程开始时间
```

在 Linux 系统中，用户、进程、内核、设备之间的关系如下。

☑ 每个用户拥有多个同时运行的进程，多个进程分别属于不同的用户。
☑ 用户进程通过系统调用接口来访问操作系统的服务。
☑ 允许多个用户同时存在并运行不同的进程。
☑ 通过设备驱动来访问硬件设备与资源，例如数据存储和网络设备等。
☑ 所有进程（无论是否属于同一用户）各自运行于独立的内存空间中。

Linux 系统通过使用 CPU 的内存管理单元（MMU）将整个系统内存分为内核区与用户区。操作系统内核本身运行于内核区，而用户进程运行于用户区。在加载进程时，操作系统需要完成如下两个操作：

☑ 将进程的可执行映像（代码和数据）映射到虚拟内存空间的用户区中。
☑ 将本身的内核代码与数据映射到虚拟地址空间的内核区。

在 Linux 系统中，通过虚拟内存管理将内核区和用户区的访问权限设置为不同的级别。其中，操作系统内核具有最高的虚拟内存访问权限，而进程在运行时能访问的存储空间只限于其可见的虚拟内存空间的用户区。在进程虚拟内存空间的用户区中，包含了进程本身的程序代码和数据，可以进一步细分为代码段、数据段、堆、栈，也可以细分为进程运行的环境变量、命令行参数传递区域等。

在 Linux 操作系统中，通过进程内存管理机制可以确保一个进程无法访问其他进程的内存空间，这样可以保证应用进程无法侵入操作系统空间和其他进程的内存空间中，通过更改代码的方式以获得更高权限，无论是恶意的还是无意的。为了实现上述安全机制，Linux 规定了进程虚拟地址与物理地址之间的映射关系，也规定了进程镜像的内存地址的分配机制。具体说明如下。

☑ 在 Linux 进程的虚拟内存内核区中，内核代码和数据被映射到内核区，这样可以确保进程在运行中得到操作系统的支持。
☑ Linux 内核区总是映射到物理内存的低地址空间。
☑ 进程有自己独立的由内核区和用户区组成的虚拟内存空间，这部分被映射到物理内存中。
☑ Linux 系统将进程虚拟内存空间内核区的访问权限设置为 0 级，用户区设置为 3 级。因为内核与进程访问虚拟内存的权限不同，所以进程代码不能访问内核区的代码与数据。
☑ 在进程虚拟内存用户区中，进程的可执行映像的代码和数据被映射到虚拟内存的用户区，也就是进程用户区由进程的程序代码和数据组成。用户区映射到物理地址内核映射区以上的任意地址空间。

14.1.6 沙箱模型

沙箱是一种安全环境，现实中的沙箱（sandbox）是一种儿童玩具，例如 KFC（肯德基）中一个装满小球的容器，孩子们可以在其中随意玩耍，同时起到了保护儿童的作用。最近几年来，随着网络安全问题的日益突出，人们将更多的将沙箱技术应用在网上冲浪领域。从技术实现角度来说，其原理是从原有的阻止可疑程序对系统访问，转换为将可疑程序对磁盘、注册表等的访问重定向到指定文件夹下，从而消除对系统的危害。本节将详细讲解 Android 系统沙箱模型的基本知识，为读者学习本书后面的知识打下基础。

1. Java 中的沙箱模型

在主流开发语言 Java 技术中，沙箱具有很重要的安全意义。为了确保 Java 技术不会被恶意目的所利用，Java 设计了一套精密的安全模型，即安全管理器（security manager），可以检查有权使用的所有系统资源。在默认的情况下，只允许无害的操作，要想允许执行其他操作，代码需得到数字签名，用户必须得到数字认证。Java 语言规定，在沙箱中的程序存在如下限制。

☑ 不能运行任何本地可执行程序。

☑ 不能从本地计算机文件系统中读取任何信息，也不能向本地计算机文件系统中写入任何信息。

☑ 不能查看除 Java 版本信息，以及少数几个无害的操作系统详细信息外的任何有关本地计算机的信息。特别需要注意的是，在沙箱中的代码不能查看用户名和 E-mail 地址等信息。

☑ 远程加载的程序不能与除下载程序所在的服务器之外的任何主机通信，这个服务器称为源主机（originating host）。这条规则通常称为“远程代码只能与家人通话”，此规则将确保用户不会被代码探查到内部网络资源（在 Java SE6 中，Java Web Start 应用程序可以与其他网络连接，但必须得到用户的同意）。

2. Android 系统中的沙箱机制

随着沙箱技术的盛行，很多主流公司的产品都采用了沙箱技术来保证上网安全，例如 360 浏览器。而对于开源的 Android 系统来说，也特意引入了这样一个安全概念。在传统的 Linux 中，一个用户 ID 识别一个给定用户。而在 Android 系统中，一个用户 ID 识别一个应用程序。应用程序在安装时被分配用户 ID，应用程序在设备上的生存期间内，用户 ID 保持不变。权限是关于允许或限制应用程序（而不是用户）访问设备资源。

Android 系统通过使用沙箱的概念来实现应用程序之间的分离和权限，以允许或拒绝一个应用程序访问设备的资源，例如，文件和目录、网络、传感器和 API。基于此，Android 使用一些 Linux 实用工具来实现应用程序被允许执行的操作，例如，进程级别的安全性、与应用程序相关的用户/组 ID 以及权限。

Android 应用程序运行在其自己的 Linux 进程上，并被分配一个唯一的用户 ID。在默认情况下，运行在基本沙箱进程中的应用程序没有被分配权限，因而防止了此类应用程序访问系统或资源。但是 Android 应用程序可以通过应用程序的 manifest 文件来请求权限。

要想允许其他应用程序可以访问 Android 应用程序的资源，可以通过如下两点来实现。

（1）声明适当的 manifest 权限。

（2）在同一进程中运行其他受信任的应用程序，从而共享对其数据和代码的访问。

在 Android 系统中，可以在相同的进程中运行不同的应用程序。此时必须先使用相同的私钥签署这些应用程序，然后使用 manifest 文件为其分配相同的 Linux 用户 ID，这样可以通过相同的“值/名”来定义 manifest 属性 android:sharedUserId 的方式实现。

在 Android 系统中，如下应用间的安全性由 Linux 操作系统的标准进程级安全机制实现。

☑ 应用程序进程之间的安全。

☑ 应用程序与操作系统之间的安全。

在默认状态下，Android 应用程序之间无法交互，运行在进程沙箱内的应用程序没有被分配权限，这样就无法访问系统或资源。所以无论是直接运行于操作系统上的应用程序，还是运行于 Dalvik VM 机的应用程序，都会得到同样的安全隔离与保护，被限制在各自“沙箱”内的应用程序互不干扰，这样做的好处是降低对系统与其他应用程序的损害。Android 应用程序通过使用沙箱机制，可以实现互相不具备信任关系的应用程序的隔离，以便独自运行。

14.1.7　Android 应用程序的安全机制

在现实应用中，Android 系统面临的安全问题主要是来自针对应用程序的攻击，病毒程序也主要是通过恶意应用程序传播的。在本节将简要讲解 Android 应用程序安全机制的基本知识。

1. AndroidManifest.xml 文件的权限机制

在安装 Android 应用程序时分配一个用户标志（UID），目的是区别于其他应用程序以保护自己的数据不被其他应用获取。在 Android 系统中，会根据不同的用户和组来分配不同权限，例如，分别设置访问网络、访问 GPS 数据等权限功能。在底层应用模块中，上述 Android 权限映射为 Linux 的用户与组权限。

在 Android 系统中，在文件 AndroidManifest.xml 中通过<permission>、<permission-group>和<permission-tree>等标签来指定应用层显式权限，设置应用程序包（.apk 文件）的权限信息。如果想要申请某个具体的权限，需要使用<uses-permission>标签来指定。在声明一个权限时，需要声明包含权限名称、属于的权限组与保护级别。

在 Android 系统中，权限组会根据权限的功能划分成不同的集合，其中又包含多个具体权限，例如，收发短信、无线上网等功能可以划入“收费数据业务”权限组。

在 Android 系统中，可以将权限的保护级别分为 Normal、Dangerous、Signature 和 Signatureorsystem 共 4 种，通过这 4 种不同的级别限定应用程序行使此权限时的认证方式，具体说明如下。

☑ Normal 级别：只要申请后就可以使用。

☑ Dangerous 级别：在安装时经用户确认才可以使用。

☑ Signature 和 Signatureorsystem 级别：应用程序必须为系统用户，如 OEM 制造商或 ODM 制造商等。

另外，如果没有在文件 AndroidManifest.xml 中声明某个框架层与系统层逐级验证权限，那么程序运行时会报错，此时可以通过命令行调试工具 logcat 查看系统日志，可发现需要某权限的提示信息。

在 Android 系统中，共享 UID 的应用程序可以与系统另一用户程序使用同一签名，也可共享同一个权限。此类机制可以通过文件在 AndroidManifest 文件中设置 sharedUserId 的方式实现，例如，通过如下代码可以获得系统权限，但是这种程序只对系统软件起作用。

```
android:sharedUserId="android.uid.shared"
```

另外，需要注意的是，从 Android 2.3 版本之后，即使有 root 权限也无法执行很多底层命令和 API。例如，su 到 root 用户，执行 ls 等命令都会播报没有权限之类的错误。

2. 发布签名机制

在开发 Android 应用程序时，开发者都必须对发布的程序设置数字签名，也就是使用私有密钥数字签署一个给定的应用程序，以便及时识别出代码的作者，并检测应用程序是否发生了改变。这样可以在相同签名的应用程序之间建立信任，从而使某些应用程序安全地实现资源共享。

在生成 Android 应用程序签名时，需要生成私有密钥与公共密钥对，用私有密钥签署公共密钥证书。虽然在 Android 应用程序商店和安装包中不会安装没有数字证书的应用，但是不需要权威机构来认证签名的数字证书。Android 应用程序签名工作可以通过第三方完成，例如，OEM 厂商、运营商及应用程序商店等，也可以由开发者自己完成签名，这就是自签名。自签名允许开发者不依赖于任何第三方自由发布应用程序。

在安装 Android 应用程序的 APK 文件时，系统安装程序会先检查 APK 是否被签名，智能安装有签名的程序。当升级该 Android 应用程序时，需要检查新版应用的数字签名与已安装的应用程序的签名是否相同。如果不相同，会被当作一个全新的应用程序来对待。在 Android 应用中，由同一个开发者设计的多个应用程序可采用同一私钥签名。具体方法是在 manifest 文件中声明共享用户的 ID，允许它们在相同的进程中运行，此时这些所属同一开发者的应用程序便可以共享代码和数据资源。Android 开发者们有可能把安装包命名为

相同的名字，通过不同的签名可以把它们区分开，保证签名不同的包不被替换掉，同时有效地防止了恶意软件替换安装的应用。

Android 提供了基于签名的权限检查，应用程序间具有相同的数字签名，它们之间可以以一种安全的方式共享代码和数据。

14.2　分区加载机制

知识点讲解：光盘:视频\视频讲解\第 14 章\分区加载机制.avi

在 Android 设备中，整个分区的类别如下。

（1）系统分区

在 Android 系统中，系统分区通常被加载为只读分区，包含操作系统内核、系统函数库、实时运行框架、应用框架与系统应用程序等。Android 系统分区由 OEM 厂商在出厂时植入，外界不能更改。当 Android 系统出现安全问题时，用户可以启动进入“安全模式”，选择只加载只读的系统分区，而不加载数据分区中的数据内容，以隔离第三方应用程序可能带来的安全威胁。

（2）Cache 分区

在 Android 系统中，Cache 分区即目录分区，在不同的目录加载不同的内容，具体说明如下。

☑ /system/app 目录：存放系统自带应用程序 APK。
☑ /system/lib 目录：存放系统库文件。
☑ /system/bin 和/system/xbin 目录：存放系统管理命令等。
☑ /system/framework 目录：存放 Android 系统应用框架的 JAR 文件。

（3）数据分区

在 Android 系统中，数据分区用于存储各类用户数据与应用程序。一般需要对数据分区设定容量限额，并且防止黑客向数据分区非法写入数据，或者防止创建非法文件对数据分区进行恶意破坏。当出现问题时，可以在“安全模式”下设置不加载数据分区，或者不启动数据分区中的应用程序。在有些情况下，甚至可以直接重新格式化数据分区，通过恢复数据的方式恢复被损坏的系统。在通常情况下，Android 数据分区加载点目录为/data，在此目录下主要包括以下子目录。

☑ /data/data 目录：保存所有 APK 程序数据。每个 APK 对应自己的 Data 目录，即在/data/data 目录下有一个与 Package 名字一样的目录。APK 只能在此目录下操作，不能访问其他 APK 的目录。
☑ /data/app 目录：保存用户安装的 APK。
☑ /data/system 目录：保存 packages.xml 、packages.list 和 appwidgets.xml 等文件，用于记录安装的软件及 Widget 信息等。
☑ /data/misc 目录：保存 Wi-Fi 账号与 VPN 设置等。

（4）SD 卡分区

在 Android 系统中，SD 卡是外置设备，可以从其他计算机系统上进行操作，完全不受 Android 系统控制。另外， SD 卡通常是 FAT 格式的文件系统，根本无法设置用户许可权限。虽然 Android 允许在加载文件系统时对整个 FAT 文件系统设置读写权限，但是不能针对 FAT 中的个别文件进行特殊操作。[1]

1. 本章的内容参考地址：http://mobile.51cto.com/netsecurity-292955.htm。

14.3 系统分析

知识点讲解：光盘:视频\视频讲解\第 14 章\系统分析.avi

从本节开始，将对 Android 版的 QQ 软件进行讲解，详细分析其运作流程和安全漏洞，并开发一个聊天记录查看器。

14.3.1 背景分析

随着 Android 系统的普及和发展，有越来越多的用户考虑手机程序的安全性问题。在此之前，国内安全论坛一直没有讨论过 Android 平台上隐私安全等话题，最多是各类短信、电话监听器的编写。这主要是由以下两点原因造成的。

（1）目前在国内的 Android 市场上此类信息窥探软件没有兴起，没有对用户利益造成大面积的伤害，因此安全厂商也无法对其作出必要的需求产品。

（2）大多数开发人员和手机自身用户对此类攻击手段并不知晓，安全意识也非常薄弱。但这些原因并不能阻碍流氓软件、病毒软件的滋生，相反，在国内市场，这些软件正处于潜伏期，一旦时机成熟，将会在移动互联网上掀起一阵“血雨腥风”。因此，尽快学习相关安全知识，加强安全意识，对安全厂商、安全爱好者以及用户都是很有必要的。

14.3.2 系统目标

QQ 是当今国内最流行的聊天软件，本章将把目标锁定在国内用户群最大的软件公司——腾讯出品的 Android 版手机 QQ 上。版本选择 2.3 版，而测试环境依然是 ROOT 过的 MOTO XT615 上。通过对其进行简单的分析后，会发现当手机获得 ROOT 权限后，可能对用户隐私带来极大的危害。

在大多数情况下，Android 手机用户在拿到新机后，通常会做如下 3 件事。

（1）水货用户会重新刷机，然后对手机进行 ROOT。而对于国行用户来说，在思考再三后也可能会 ROOT 掉手机（Android 手机软件运行在普通用户权限，程序对系统本身没有控制权。而所谓“ROOT 掉手机”就是通过一些手段让 Android 手机可拥有 ROOT 权限，之后软件通过“ROOT 权限”可以随意修改、访问手机中所有资源）。

（2）将手机 ROM 中自带的广告软件、无用软件精减掉对手机进行优化。

（3）安装 QQ，因为这是一款国内最流行的社交软件。

本章的实例必须在拥有 ROOT 权限下的 Android 手机中才能完全运行，这在大多数环境下是成立的。

14.4 反汇编分析

知识点讲解：光盘:视频\视频讲解\第 14 章\反汇编分析.avi

手机 QQ 的界面很漂亮，UI 的布局也很人性化，如图 14-3 所示。但是在使用时发现了一个小问题，在手机无网络时打开群聊天窗口，程序也会卡死，估计程序是在消息同步时被阻塞了。





图 14-3　Android 版本的 QQ

使用 APK 反编译工具将 QQ 2.3 版本安装包进行反汇编解包。按照惯例，先打开里面的设置文件 AndroidMenifest.xml，并寻找程序启动部分，首先找到.activity.SplashActivity 文件，从名字上看为启动时的闪屏，找到 Activity 文件（位于\smali\com\tencent\mobileqq\activity 文件夹下）并打开，直接看 OnCreate()方法，具体代码如下所示。

```
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 4
    .parameter
    .prologue
    .line 41
    invoke-super {p0, p1}, Lcom/tencent/mobileqq/app/BaseActivity;->onCreate(Landroid/os/Bundle;)V
    #这里的父类也是 TX 自己编写的
    .line 42
    sput-object p0, Lcom/tencent/mobileqq/activity/SplashActivity;->instance:Lcom/tencent/mobileqq/activity/SplashActivity;
    .line 43
    invoke-virtual {p0}, Lcom/tencent/mobileqq/activity/SplashActivity;->getIntent()Landroid/content/Intent;
    .line 44
    const/4 v0, 0x1
    invoke-virtual {p0, v0}, Lcom/tencent/mobileqq/activity/SplashActivity;->requestWindowFeature(I)Z
    .line 45
    invoke-virtual {p0}, Lcom/tencent/mobileqq/activity/SplashActivity;->getWindow()Landroid/view/Window;
    move-result-object v0
    const/16 v1, 0x400
    invoke-virtual {v0, v1}, Landroid/view/Window;->addFlags(I)V
    .line 46
    const v0, 0x7f03007f
    invoke-virtual {p0, v0}, Lcom/tencent/mobileqq/activity/SplashActivity;->setContentView(I)V
    #设置要显示的 View
    .line 47
    new-instance v0, Landroid/os/Handler;   #new 一个 Handler
    invoke-direct {v0}, Landroid/os/Handler;-><init>()V
    iput-object v0, p0, Lcom/tencent/mobileqq/activity/SplashActivity;->a:Landroid/os/Handler;   #保存 Handler
    .line 52
    iget-object v0, p0, Lcom/tencent/mobileqq/activity/SplashActivity;->a:Landroid/os/Handler;
    iget-object v1, p0, Lcom/tencent/mobileqq/activity/SplashActivity;->a:Ljava/lang/Runnable;
    const-wide/16 v2, 0x7d0 #延迟的时间间隔
    invoke-virtual {v0, v1, v2, v3}, Landroid/os/Handler;->postDelayed(Ljava/lang/Runnable;J)Z
    #延迟启动一个线程进行登录操作
    .line 54
```

```
    invoke-virtual
{p0}, Lcom/tencent/mobileqq/activity/SplashActivity;->getApplication()Landroid/app/Application;  //获取 Application
    move-result-object p0
    check-cast p0, Lcom/tencent/mobileqq/app/QQApplication;
    .line 55
    invoke-static {p0}, Lcom/tencent/mobileqq/log/ExceptionHandler;->register(Lcom/tencent/mobileqq/app/QQApplication;)V
    #注册 QQApplication 的异常处理器
    .line 56
    invoke-virtual {p0}, Lcom/tencent/mobileqq/app/QQApplication;->a()Lcom/tencent/qphone/base/remote/SimpleAccount;
    move-result-object v0   #调用 a()方法并比较结果，用户是否获取成功
    if-eqz v0, :cond_0
    invoke-virtual {p0}, Lcom/tencent/mobileqq/app/QQApplication;->a()Lcom/tencent/qphone/base/remote/SimpleAccount;
    move-result-object v0
    invoke-virtual {v0}, Lcom/tencent/qphone/base/remote/SimpleAccount;->getSid()Ljava/lang/String;
    move-result-object v0 #getSid()#获取 SID 自动登录字符串
    if-eqz v0, :cond_0
    .line 58
    invoke-virtual {p0}, Lcom/tencent/mobileqq/app/QQApplication;->a()Lcom/tencent/qphone/base/remote/SimpleAccount;
    move-result-object v0
    invoke-virtual {v0}, Lcom/tencent/qphone/base/remote/SimpleAccount;->getUin()Ljava/lang/String;
    move-result-object v0
    invoke-static {v0}, Lcom/tencent/mobileqq/log/ReportLog;->setUserUin(Ljava/lang/String;)V
    .line 59
   invoke-virtual {p0}, Lcom/tencent/mobileqq/app/QQApplication;->a()Lcom/tencent/qphone/base/remote/SimpleAccount;
    move-result-object v0
    invoke-virtual {v0}, Lcom/tencent/qphone/base/remote/SimpleAccount;->getSid()Ljava/lang/String;
    move-result-object v0
    invoke-virtual {v0}, Ljava/lang/String;->getBytes()[B
    move-result-object v0
    invoke-static {v0}, Lcom/tencent/mobileqq/log/ReportLog;->setSig([B)V
    #ReportLog 记录结果，到这里用户就可以算是自动登录了
    .line 65
    :cond_0
    invoke-static { }, Lcom/tencent/qphone/base/util/LoginHelper;->getCommonConfig()Ljava/lang/String;
    move-result-object v0 #通过 JNI 调用获取通用配置信息
    .line 66
    const-string v1, "<QQIniUrl>"
    invoke-virtual {v0, v1}, Ljava/lang/String;->indexOf(Ljava/lang/String;)I
    move-result v1
    .line 67
    const-string v2, "</QQIniUrl>"
    invoke-virtual {v0, v2}, Ljava/lang/String;->indexOf(Ljava/lang/String;)I
    move-result v2
    .line 69
    if-ltz v1, :cond_1
    if-le v2, v1, :cond_1
    .line 70
    const-string v3, "<QQIniUrl>"
    invoke-virtual {v3}, Ljava/lang/String;->length()I
    move-result v3
    add-int/2addr v1, v3
```

```
    invoke-virtual {v0, v1, v2}, Ljava/lang/String;->substring(II)Ljava/lang/String;
    move-result-object v0
    invoke-virtual {v0}, Ljava/lang/String;->trim()Ljava/lang/String;
    move-result-object v0
    .line 71
    sput-object v0, Lcom/tencent/mobileqq/log/ReportLog;->URL_LOG_UPLOAD:Ljava/lang/String;
    .line 72
   invoke-virtual {p0}, Lcom/tencent/mobileqq/app/QQApplication;->a()Lcom/tencent/mobileqq/
utils/httputils/HttpCommunicator;
    move-result-object v0
    invoke-static {v0, p0}, Lcom/tencent/mobileqq/log/ReportLog;->upload(Lcom/tencent/mobileqq/utils/httputils
/HttpCommunicator;Landroid/content/Context;)V
    .line 74
    :cond_1
    const/4 v0, 0x0
    const-string v1, "SplashActivity onCreate()"
    invoke-static {v0, v1}, Lcom/tencent/mobileqq/log/ReportLog;->appendLog(Ljava/lang/String;Ljava/lang/
String;)V
    .line 81
    return-void
.end method
```

在上述 OnCreate()方法的实现代码中，首先创建了一个 Handler 对象，在 Handler 中加入一个新线程。其实 Handler 与主线程使用同一线程，所以在这里主线程就转去执行 Run()方法了，新线程的代码在文件 ly.smali 中实现，打开这个文件看 Run()方法的实现，具体代码如下。

```
.method public final run()V
    .locals 5
    .prologue
    const-wide/16 v3, 0x0
    .line 107
    sget-object v0, Lcom/tencent/mobileqq/activity/NotificationActivity;->instance:Lcom/tencent/mobileqq/
activity/NotificationActivity;
    if-nez v0, :cond_2 #判断 NotificationActivity 实例是否存在，此处就是在判断 QQ 是否已经运行
    iget-object v0, p0, Lly;->a:Lcom/tencent/mobileqq/activity/SplashActivity; #获取 SplashActivity 对象
    invoke-static {v0}, Lcom/tencent/mobileqq/activity/SplashActivity;->access$000(Lcom/tencent/mobileqq/
activity/SplashActivity;)Lcom/tencent/mobileqq/app/QQApplication; #调用 access$000()方法，此处判断
QQApplication 的实例 App 是否存在
    move-result-object v0
    iget-boolean v0, v0, Lcom/tencent/mobileqq/app/QQApplication;->c:Z #app->c
    if-nez v0, :cond_2
    .line 108
    iget-object v0, p0, Lly;->a:Lcom/tencent/mobileqq/activity/SplashActivity;
    const-string v1, "mobileQQ"
    const/4 v2, 0x0
    invoke-virtual {v0, v1, v2}, Lcom/tencent/mobileqq/activity/SplashActivity;->getSharedPreferences(Ljava/
lang/String;I)Landroid/content/SharedPreferences; #getSharedPreferences 是获取 QQ 目录 SharedPreference
配置信息
    move-result-object v0
    .line 110
    const-string v1, "firstTimeRun"
    invoke-interface {v0, v1, v3, v4}, Landroid/content/SharedPreferences;->getLong(Ljava/lang/String;J)J
```

```
    move-result-wide v1 #获取首次启动时间
    .line 112
    cmp-long v1, v1, v3 #判断配置文件中的启动时间是否为空
    if-nez v1, :cond_0
    .line 113
    invoke-interface {v0}, Landroid/content/SharedPreferences;->edit()Landroid/content/SharedPreferences$
Editor; #准备写入数据
    move-result-object v0
    .line 114
    const-string v1, "firstTimeRun"
    const-wide/16 v2, 0x1
    invoke-interface {v0, v1, v2, v3}, Landroid/content/SharedPreferences$Editor;->putLong(Ljava/lang/String;J)
Landroid/content/SharedPreferences$Editor;
    #写入首次启动时间
    .line 115
    invoke-interface {v0}, Landroid/content/SharedPreferences$Editor;->commit()Z #提交修改
    .line 116
    iget-object v0, p0, Lly;->a:Lcom/tencent/mobileqq/activity/SplashActivity;
    invoke-static {v0}, Lcom/tencent/mobileqq/activity/SplashActivity;->access$100(Lcom/tencent/mobileqq/
activity/SplashActivity;)V
    .line 118
    :cond_0
    iget-object v0, p0, Lly;->a:Lcom/tencent/mobileqq/activity/SplashActivity;
    invoke-virtual {v0}, Lcom/tencent/mobileqq/activity/SplashActivity;->finish()V #结束闪屏
    .line 119
    new-instance v0, Landroid/content/Intent;
    iget-object v1, p0, Lly;->a:Lcom/tencent/mobileqq/activity/SplashActivity;
    const-class v2, Lcom/tencent/mobileqq/activity/HomeActivity;
    invoke-direct {v0, v1, v2}, Landroid/content/Intent;-><init>(Landroid/content/Context;Ljava/lang/Class;)V
    #构造一个 HomeActivity 对象
 .line 120
    iget-object v1, p0, Lly;->a:Lcom/tencent/mobileqq/activity/SplashActivity;
    invoke-virtual {v1}, Lcom/tencent/mobileqq/activity/SplashActivity;->getIntent()Landroid/content/Intent;
    move-result-object v1
    const-string v2, "selfuin"
    invoke-virtual {v1, v2}, Landroid/content/Intent;->getStringExtra(Ljava/lang/String;)Ljava/lang/String;
    move-result-object v1
    .line 121
    invoke-static {v1}, Landroid/text/TextUtils;->isEmpty(Ljava/lang/CharSequence;)Z
    move-result v1
    if-nez v1, :cond_1
    .line 123
    iget-object v1, p0, Lly;->a:Lcom/tencent/mobileqq/activity/SplashActivity;
    invoke-virtual {v1}, Lcom/tencent/mobileqq/activity/SplashActivity;->getIntent()Landroid/content/Intent;
    move-result-object v1
    invoke-virtual {v0, v1}, Landroid/content/Intent;->putExtras(Landroid/content/Intent;)Landroid/content/Intent;
    .line 125
    :cond_1
    iget-object v1, p0, Lly;->a:Lcom/tencent/mobileqq/activity/SplashActivity;
    invoke-virtual   {v1,   v0},   Lcom/tencent/mobileqq/activity/SplashActivity;->startActivity(Landroid/content/
Intent;)V #启动 HomeActivity
```

```
    .line 128
    :cond_2
    return-void
.end method
```

上述线程实现代码的功能是获取并保存程序首次启动时间，然后启动 HomeActivity。接下来直接看 HomeActivity 中的 OnCreate()方法，其整个过程是验证 QQ 自动登录，如果在此期间发生了错误或自动登录验证失败，则会执行如下代码。

```
    .line 166
    :cond_a
    new-instance v0, Landroid/content/Intent;
    const-string v1, "com.tencent.mobileqq.action.LOGIN"
    invoke-direct {v0, v1}, Landroid/content/Intent;-><init>(Ljava/lang/String;)V
    const/high16 v1, 0x4
    invoke-virtual {v0, v1}, Landroid/content/Intent;->addFlags(I)Landroid/content/Intent;
    move-result-object v0
    const/16 v1, 0x3e8
    invoke-virtual {p0, v0, v1}, Lcom/tencent/mobileqq/activity/HomeActivity;->startActivityForResult(Landroid/
content/Intent;I)V
    goto :goto_3
```

在文件 AndroidMenifest.xml 中可以发现 com.tencent.mobileqq.action.LOGIN 对应的是 LoginActivity，打开文件 LoginActivity.smali 直接浏览登录按钮监听器的代码，其中的关键代码如下。

```
.method public onClick(Landroid/content/DialogInterface;I)V
    .locals 5
    .parameter
    .parameter
    .prologue
    const/4 v1, -0x1
    .line 811
    iget v0, p0, Lcom/tencent/mobileqq/activity/LoginActivity;->a:I
    if-eq v0, v1, :cond_3

    ...
    .line 836
    :cond_2
    invoke-virtual {v0}, Lhf;->notifyDataSetChanged()V
    .line 837
    iget-object v0, p0, Lcom/tencent/mobileqq/activity/LoginActivity;->e:Landroid/widget/CheckBox;
    invoke-virtual {v0}, Landroid/widget/CheckBox;->isChecked()Z
    move-result v0
    if-eqz v0, :cond_3
    .line 838
    new-instance v0, Lcom/tencent/mobileqq/data/QQEntityManagerFactory;
    invoke-direct {v0, v2}, Lcom/tencent/mobileqq/data/QQEntityManagerFactory;-><init>(Ljava/lang/String;)V
    .line 839
    invoke-virtual {v0, v2}, Lcom/tencent/mobileqq/persistence/EntityManagerFactory;->build(Ljava/lang/
String;)Landroid/database/sqlite/SQLiteOpenHelper;
    move-result-object v1
    invoke-virtual {v1}, Landroid/database/sqlite/SQLiteOpenHelper;->getWritableDatabase()Landroid/
database/sqlite/SQLiteDatabase;
    move-result-object v1
```

```
    .line 840
    const-string v2, "select name from sqlite_master where type=\"table\" and name like \"mr_%\""
    const/4 v3, 0x0
    invoke-virtual {v1, v2, v3}, Landroid/database/sqlite/SQLiteDatabase;->rawQuery(Ljava/lang/String;[Ljava/
lang/String;)Landroid/database/Cursor;
    move-result-object v2
    .line 844
    :goto_0
    invoke-interface {v2}, Landroid/database/Cursor;->moveToNext()Z
    move-result v3
    if-eqz v3, :cond_4
    .line 845
    const/4 v3, 0x0
    invoke-interface {v2, v3}, Landroid/database/Cursor;->getString(I)Ljava/lang/String;
    move-result-object v3
    .line 846
    invoke-static {v3}, Lcom/tencent/mobileqq/persistence/TableBuilder;->dropSQLStatement(Ljava/
lang/String;)Ljava/lang/String;
    move-result-object v3
    invoke-virtual {v1, v3}, Landroid/database/sqlite/SQLiteDatabase;->execSQL(Ljava/lang/String;)V
    goto :goto_0
    :catch_0
    move-exception v0
    .line 856
    :cond_3
    :goto_1
    return-void
    .line 848
    :cond_4
    new-instance v2, Lcom/tencent/mobileqq/data/RecentUser;
    invoke-direct {v2}, Lcom/tencent/mobileqq/data/RecentUser;-><init>()V
    invoke-virtual {v2}, Lcom/tencent/mobileqq/data/RecentUser;->getTableName()Ljava/lang/String;
    move-result-object v2
    invoke-static {v2}, Lcom/tencent/mobileqq/persistence/TableBuilder;->dropSQLStatement(Ljava/lang/
String;)Ljava/lang/String;
    move-result-object v2
    invoke-virtual {v1, v2}, Landroid/database/sqlite/SQLiteDatabase;->execSQL(Ljava/lang/String;)V
    .line 850
    invoke-virtual {v0}, Lcom/tencent/mobileqq/persistence/EntityManagerFactory;->close()V
    :try_end_0
    .catch Ljava/lang/Exception; {:try_start_0 .. :try_end_0} :catch_0
    goto :goto_1
.end method
```

在上述代码中，执行了一条 SQL 语句 select name from sqlite_master where type=\"table\" and name like \"mr_%\"，功能是查询表中所有以 mr_开头的表名，这与在 Sqlite 命令行下输入.tables mr_%的作用是一样的。由此可见，TX 保存信息使用的是 Sqlite。

在 Android 系统中存储信息的方式有以下几类。

（1）第 1 类存储为普通的文件存储，通过 Java 提供的 I/O 库对文件读取与写入，这与其他的语言或平台是一样的。显然，由于直接将数据暴露给用户，这种存储机制的安全性是最低的。



（2）第 2 类是 Android 中提供的一个私有数据存储类 SharedPerferences，该类提供了简单的键值对来对数据进行存储，例如，360 手机卫士就是用这种方法保存数据，这个类存储的数据只能被软件自身访问（ROOT 过的手机除外），加强了数据安全性。

（3）第 3 类是 SQLite 数据库存储，这是一个轻量级的关系型数据库，现已在各大平台上广泛应用，在 Android 系统中作为一个标准数据库的存在。

（4）第 4 类是 ContentProvider 方式存储，第 2 类与第 3 类存储的数据都是程序私有的，然而新的问题是两个或多个程序有时候可能需要进行数据交换，如同一个公司产品的无缝结合。解决这个问题要靠 ContentProvider，一个 ContentProvider 接口实现了一套标准的方法接口，应用程序可以通过实现 ContentProvider 接口将自己的数据暴露出去，这样就做到了选择性的数据开放。

（5）第 5 类是网络存储，也就是通过网络实现对数据的存储与获取，此类存储一般作为软件的附加功能。

在 Android 系统中，SQLite 数据库一般保存在/data/data/<package_name>/databases/目录下，<package_name>为软件的包名。如果通过 adb shell 或 DDMS 命令进入手机/data/data/com.tencent.mobileqq/databases/目录，会发现里面有好几个以.db 结尾的文件（前提是手机必须 ROOT，否则是无法查看的），并且其中一个是以自己 QQ 号命名的。

接下来试着手动查看一下这个 DB 文件的信息，具体操作过程如下。

（1）打开 CMD 命令行工具，首先输入 chcp 65001，会弹出如图 14-4 所示的界面。

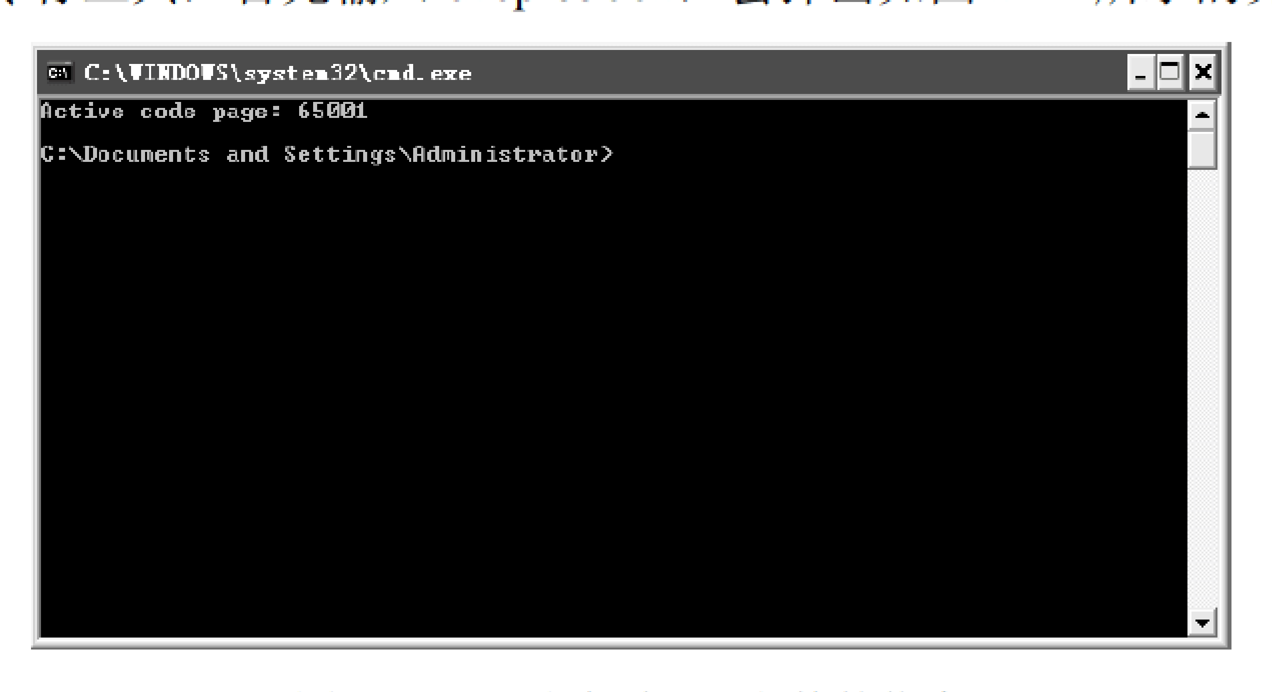


图 14-4　手动查看 DB 文件的信息

（2）在 CMD 标题栏上右击并选择“属性”命令，设置字体为 Lucida Console，如图 14-5 所示。



图 14-5　设置字体为 Lucida Console

做这两步工作是将字符集从 ANSI 变成 UTF-8，否则在显示中文字符时会呈现乱码。

（3）导出以自己名字命名的 DB 文件（在笔者的手机上，没有 Sqlite3 程序），直接将 DB 文件导出后用 sqlite3.exe（这个文件包含在 Android SDK 中，运行前先配置好环境变量）打开，如图 14-6 所示。

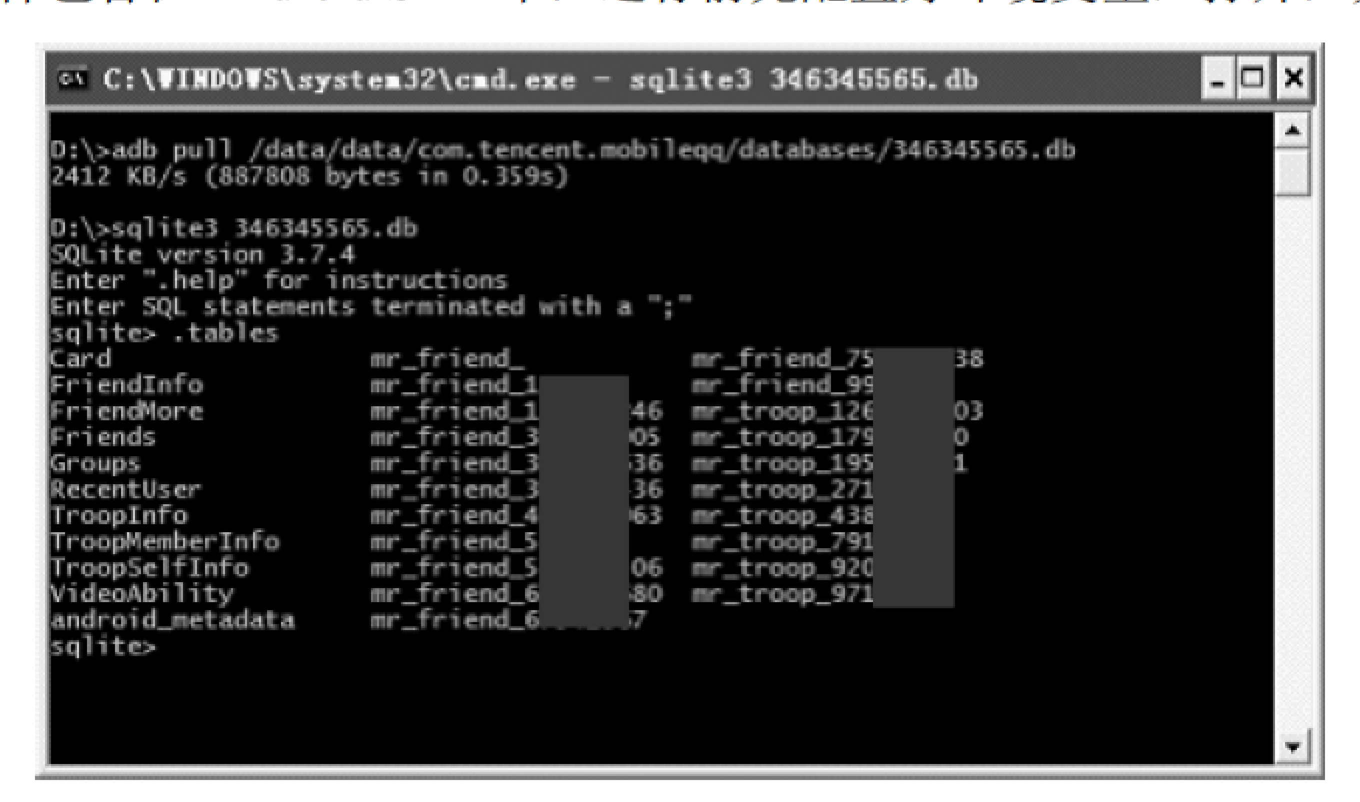


图 14-6　导出 DB 文件

（4）列举出数据库中含有的表名，查看其中 Friends 表中的字段信息，如图 14-7 所示。

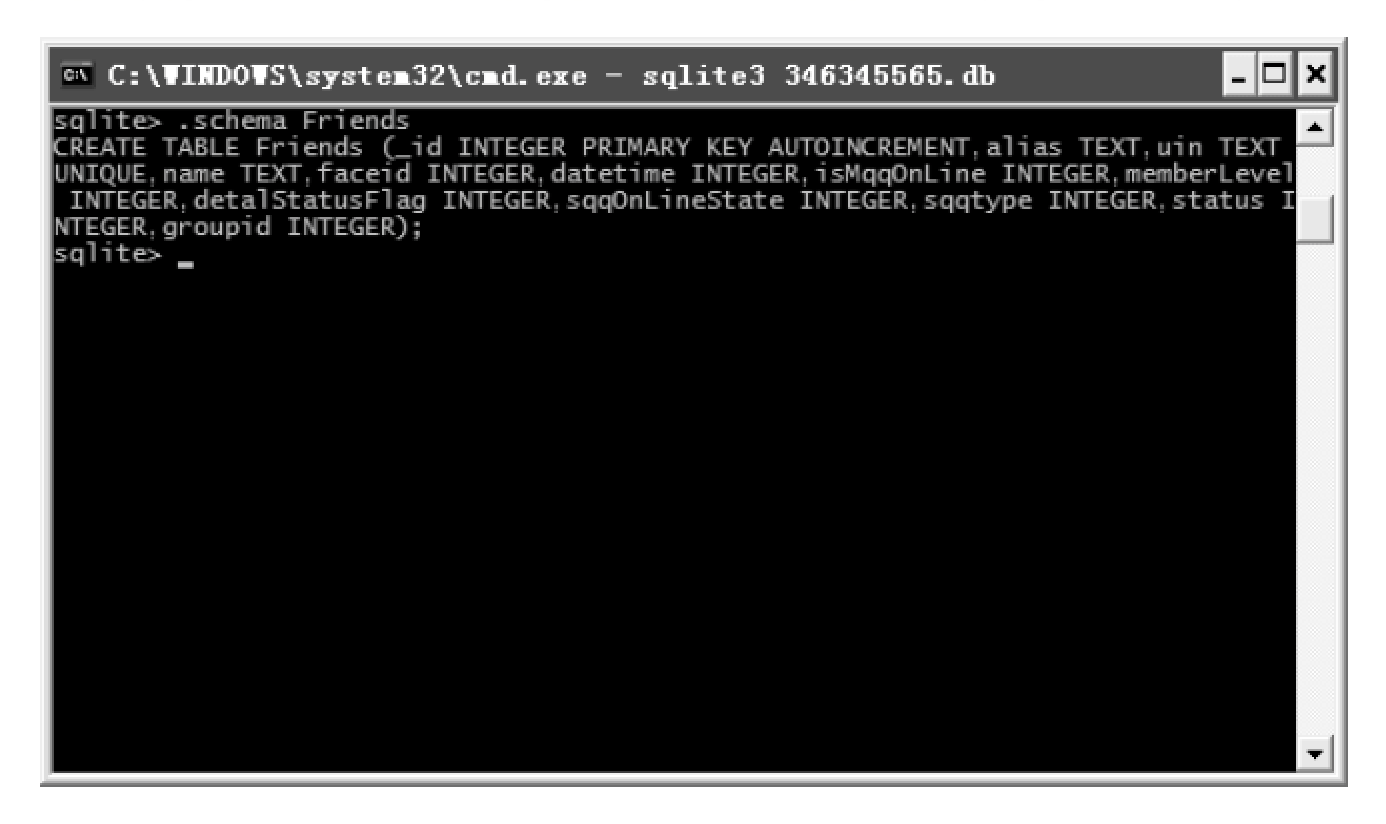


图 14-7　查看 Friends 表中的字段信息

（5）通过对字段的猜测执行 SQL 语句“select uin, name, groupid from Friends;”，查询结果如图 14-8 所示。

```
C:\WINDOWS\system32\cmd.exe - sqlite3 346345565.db
sqlite>
sqlite> .schema Friends
CREATE TABLE Friends (_id INTEGER PRIMARY KEY AUTOINCREMENT,alias TEXT,uin TEXT
UNIQUE,name TEXT,faceid INTEGER,datetime INTEGER,isMqqOnLine INTEGER,memberLevel
 INTEGER,detalStatusFlag INTEGER,sqqOnLineState INTEGER,sqqtype INTEGER,status I
NTEGER,groupid INTEGER);
sqlite> select uin, name, groupid from Friends;
[redacted]
```

图 14-8　查询结果



（6）此时可以发现，QQ 中的好友都被列举出来了。接下来的工作就是分析每个表的含义，然后构造 SQL 语句来获取数据，分析出的数据如表 14-1 所示。

表 14-1　表信息

表　名	存储的数据
Card	可能是身份信息
FriendInfo	好友的信息
FriendMore	更多好友 ，最新的说说与好友 QQ 号
Friends	好友列表
Groups	好友分组
RecentUser	最新好友
TroopInfo	QQ 群
TroopMemberInfo	QQ 群成员信息
TroopSelfInfo	QQ 群自身信息
VideoAbility	有摄像头的好友
mr_friend_XXX	好友聊天记录（XXX 为好友 QQ 号）
mr_troop_XXX	群聊天记录（XXX 为 QQ 群号）

14.5　实现 QQ 聊天记录查看器系统

知识点讲解：光盘:视频\视频讲解\第 14 章\实现 QQ 聊天记录查看器系统.avi

本节将通过一个具体实例的实现过程，详细讲解在 Android 系统中开发一个 QQ 聊天记录查看器系统。本节开发的 QQ 聊天记录查看器系统是完全基于本章前面 4 节的理论和反编译内容为基础实现的。

14.5.1　系统架构

本系统程序首先运行到 QQ 相关的目录获取数据库，然后复制到自身目录下，通过 Android 提供的 Sqlite 操作类对数据进行解析，构造不同的页面并显示出来。程序需要显示出本地 QQ 的好友列表、群列表、QQ 消息、群消息，可分页显示消息，因此需要以下的几个操作类。

- ☑ QQFriendService 类负责 QQ 好友的操作，包括获取好友列表、好友 QQ 号、好友分组。
- ☑ QQTroopService 类负责群的操作，包括获取群列表、群号、群的备注信息。
- ☑ QQMessageService 类负责 QQ 好友与群的消息操作，查询并显示聊天消息。

PageUtil 类负责聊天消息的分页查询。

完整的系统目录结构如图 14-9 所示。

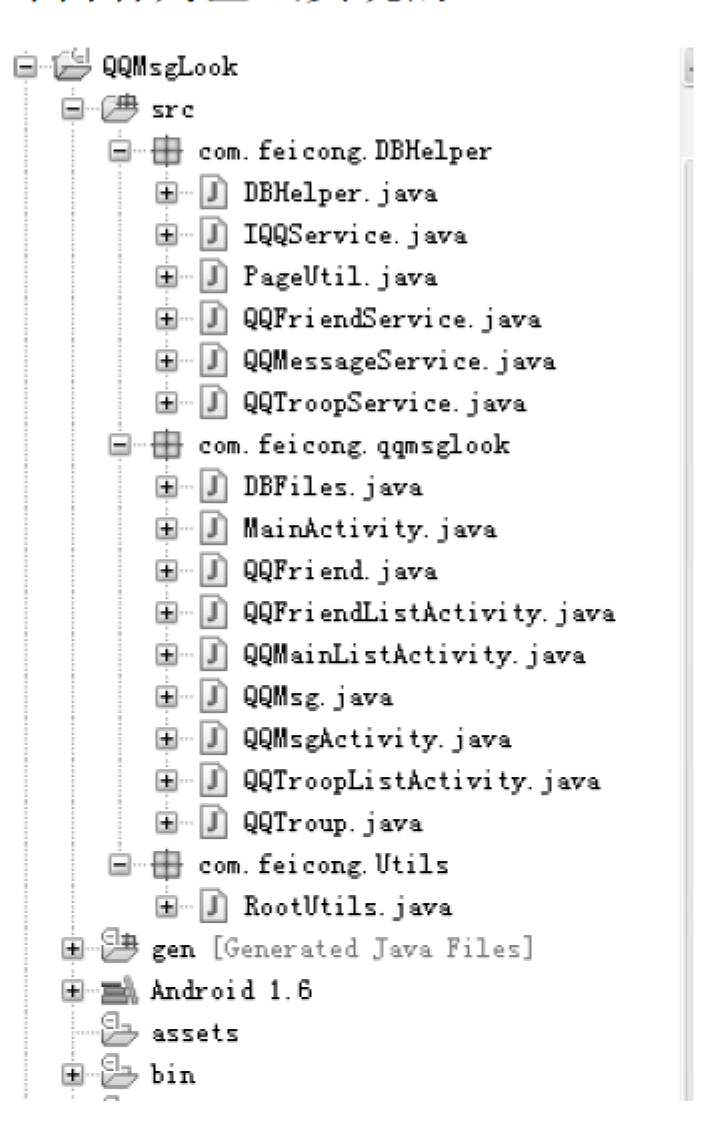


图 14-9　系统目录结构

14.5.2 实现公共类

（1）因为 3 个 Service 类都要有查询列表、数目等操作，所以可以抽象出一个 IQQService 接口。实现文件 IQQService.java 的具体代码如下。

```
public interface IQQService{
    public List<Object> queryList(int startIndex, int endIndex);
    public long queryCount();
    public ArrayList<HashMap<String, String>> setAdapterListData(int startIndex, int endIndex);
}
```

方法 etAdapterListData()的功能是返回 ListView 的列表项数据。

（2）PageUtil 类的功能是调用 IQQService 的几个操作方法来返回数据，具体实现代码如下所示。

```
public class PageUtil {
    public static int pagesize = 20;
    public int currentpage;
    private IQQService service;
    private int count;
    private int pagecount;

    public PageUtil(IQQService service){
        super();
        this.service = service;
        currentpage = 1;
        count = new Long(service.queryCount()).intValue();
        pagecount = count / pagesize;
        if(count % pagesize != 0) pagecount++;
        Log.i("QQMsgLook", String.valueOf(currentpage) + ' '
                + String.valueOf(count) + ' '
                + String.valueOf(pagecount));
    }
    public int getPagecount(){
        return pagecount;
    }
    public int getCount(){
        return count;
    }
    public int getCurrentpage(){
        return currentpage;
    }
    public void firstPage(){
        currentpage = 1;
    }
    public void nextPage(){
        currentpage++;
    }
    public void prevPage(){
        currentpage--;
    }
    public void endPage(){
        currentpage = pagecount;
```

```
	}
	public void setCurrentpage(int currentpage){
		this.currentpage = currentpage;
	}
	public ArrayList<HashMap<String, String>> getList() {
		if (currentpage == pagecount) {
			return service.setAdapterListData((pagecount - 1) * pagesize, pagesize);
		} else if (currentpage == 1){
			return service.setAdapterListData(0, pagesize);
		}else {
			return service.setAdapterListData((currentpage - 1) * pagesize, pagesize);
		}
	}
}
```

（3）服务类 QQFriendService、QQMessageService 和 QQTroopService 的实现方法类似，只是执行的 SQL 语句不同。其中，QQFriendService 类的具体实现代码如下。

```
/**
 * 此类查询 QQ 相关信息
 * @author Administrator
 *
 */
public class QQFriendService implements IQQService {
	private DBHelper db;
	public QQFriendService(DBHelper db) {
		this.db = db;
		Log.i("QQFriendService", "QQFriendService Create...");
	}
	/**
	 * 查找好友
	 * @param qqUin
	 * @return
	 */
	public QQFriend find(String qqUin) {
		try
		{
			SQLiteDatabase data = db.getReadableDatabase();
			StringBuilder sb = new StringBuilder(
					"select group_name,name from Friends,Groups where Friends.groupid=Groups.
group_id and uin=");
			sb.append(qqUin);
			Log.i("SQL", sb.toString());
			Cursor cursor = data.rawQuery(sb.toString(), null);
			if (cursor.moveToNext())
			{
				return new QQFriend(cursor.getString(0), cursor.getString(1),
						qqUin);
			}
		} catch (Exception e)
		{
			e.printStackTrace();
```

```
            }
            return null;
        }

        public List<Object> queryList(int startIndex, int endIndex)
        {
            List<Object> friends = new ArrayList<Object>();
            try
            {
                SQLiteDatabase data = db.getReadableDatabase();
                StringBuilder sb = new StringBuilder(
                        "select group_name, name, uin from Friends,Groups where Friends.groupid=Groups.
group_id order by group_id");
                sb.append(" limit ");
                sb.append(String.valueOf(startIndex));
                sb.append(',');
                sb.append(String.valueOf(endIndex));
                Log.i("SQL", sb.toString());
                Cursor cursor = data.rawQuery(sb.toString(), null);
                while (cursor.moveToNext())
                {
                    Log.i("SQL", cursor.getString(1));
                    friends.add(new QQFriend(cursor.getString(0), cursor
                            .getString(1), cursor.getString(2)));
                }
            } catch (Exception e)
            {
                e.printStackTrace();
            }
            return friends;
        }
        /**
         * 查询好友数目
         */
        public long queryCount()
        {
            SQLiteDatabase data = db.getReadableDatabase();
            Cursor cursor = data.rawQuery("select count(*) from Friends", null);
            if (cursor.moveToNext()) {
                return cursor.getLong(0);
            }
            return 0;
        }
        public ArrayList<HashMap<String, String>> setAdapterListData(int startIndex, int endIndex)
        {
            List<Object> objs;
            ArrayList<HashMap<String, String>> list = new ArrayList<HashMap<String, String>>();
            try
            {
                objs = queryList(startIndex, endIndex);
                for (Object obj : objs)
```

```
            {
                HashMap<String, String> map = new HashMap<String, String>();
                QQFriend friend = (QQFriend) obj;
                map.put("memberLevel", friend.getMemberLevel());
                map.put("name", friend.getName());
                map.put("qquin", friend.getqqUin());
                list.add(map);
            }
        } catch (Exception e)
        {
            e.printStackTrace();
        }
        return list;
    }
}
```

QQMessageService 类的具体实现代码如下。

```
public class QQMessageService implements IQQService{
    private DBHelper db;
    private String uin;
    private Boolean bQQ;

    public QQMessageService(DBHelper db, String uin, Boolean bQQ){
        this.db = db;
        this.uin = uin;
        this.bQQ = bQQ;
        Log.i("QQMessageService", "QQMessageService Create...");
    }

    public ArrayList<HashMap<String, String>> setAdapterListData(int startIndex, int endIndex){
    List<Object> msgs;
    ArrayList<HashMap<String, String>> list = new ArrayList<HashMap<String, String>>();
        msgs = queryList(startIndex, endIndex);
        for (Object obj : msgs){
          HashMap<String, String> map = new HashMap<String, String>();
          QQMsg msg = (QQMsg)obj;
          map.put("uin", msg.getUin());
          map.put("msg", msg.getMessage());
          list.add(map);
        }
        return list;
    }

    /**
     * 查询好友或群消息
     */
    public List<Object> queryList(int startIndex, int endIndex)
    {
        List<Object> messages = new ArrayList<Object>();
        SQLiteDatabase data = db.getReadableDatabase();
        StringBuilder sb = new StringBuilder("select senderuin,msg from mr_");
        if(bQQ){
```

```
				sb.append("friend_");
			}
			else{
				sb.append("troop_");
			}
			sb.append(uin);
			sb.append(" limit ");
			sb.append(String.valueOf(startIndex));
			sb.append(',');
			sb.append(String.valueOf(endIndex));
			Log.i("SQL", sb.toString());
			try
			{
				Cursor cursor = data.rawQuery(sb.toString(), null);
				while (cursor.moveToNext())
				{
					messages.add(new QQMsg(cursor.getString(0), cursor.getString(1)));
				}
			} catch (Exception e)
			{
				// TODO: handle exception
			}
			return messages;
		}
		/**
		 * 查询消息数目
		 */
		public long queryCount()
		{
			SQLiteDatabase data = db.getReadableDatabase();
			StringBuilder sb = new StringBuilder("select count(*) from mr_");
			if(bQQ){
				sb.append("friend_");
			}
			else{
				sb.append("troop_");
			}
			sb.append(uin);
			Cursor cursor = data.rawQuery(sb.toString(), null);
			if (cursor.moveToNext()) {
				return cursor.getLong(0);
			}
			return 0;
		}
}
```

QQTroopService 类的具体实现代码如下。

```
public class QQTroopService implements IQQService {
		private DBHelper db;
		public QQTroopService(DBHelper db) {
			this.db = db;
			Log.i("QQTroopService", "QQTroopService Create...");
```

```
    }

    public List<Object> queryList(int startIndex, int endIndex)
    {
        List<Object> troops = new ArrayList<Object>();
        try
        {
            SQLiteDatabase data = db.getReadableDatabase();
            StringBuilder sb = new StringBuilder("select troopname, troopuin, troopmemo from TroopInfo");
            sb.append(" limit ");
            sb.append(String.valueOf(startIndex));
            sb.append(',');
            sb.append(String.valueOf(endIndex));
            Log.i("SQL", sb.toString());
            Cursor cursor = data.rawQuery(sb.toString(), null);
            while (cursor.moveToNext())
            {
                Log.i("SQL", cursor.getString(0));
                troops.add(new QQTroup(cursor.getString(0), cursor.getString(1), cursor.getString(2)));
            }
        } catch (Exception e)
        {
            e.printStackTrace();
        }
        return troops;
    }
    /**
     *查询群数目
     */
    public long queryCount()
    {
        try
        {
            SQLiteDatabase data = db.getReadableDatabase();
            Cursor cursor = data.rawQuery("select count(*) from TroopInfo",
                    null);
            if (cursor.moveToNext())
            {
                return cursor.getLong(0);
            }
        } catch (Exception e)
        {
            e.printStackTrace();
        }
        return 0;
    }

    public ArrayList<HashMap<String, String>> setAdapterListData(int startIndex, int endIndex)
    {
        List<Object> objs;
        ArrayList<HashMap<String, String>> list = new ArrayList<HashMap<String, String>>();
```

```
            try
            {
                objs = queryList(startIndex, endIndex);
                for (Object obj : objs)
                {
                    HashMap<String, String> map = new HashMap<String, String>();
                    QQTroup troop = (QQTroup) obj;
                    map.put("troopName", troop.getName());
                    map.put("troopUin", troop.getUin());
                    map.put("troopMemo", troop.getMemo());
                    list.add(map);
                }
            } catch (Exception e)
            {
                e.printStackTrace();
            }
            return list;
        }
}
```

因为数据库操作存在各种异常，所以在上述代码中，在操作前都使用了 try…catch 语句来提高程序的健壮性，如果有 exception 异常，就用 e.printStackTrace()进行处理。

14.5.3 实现主界面

系统主界面 Activity 是 MainActivity，其布局实现文件是 main_layout.xml，功能是使用 ListView 控件列表显示在当前手机中登录过的 QQ 号码，具体实现代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
     android:id="@+id/linearlayout_main"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#dddeeeff"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textColor="#0000ff"
        android:textSize="18.0dp"
        android:text="@string/qqlist" >
    </TextView>
    <ListView
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
          android:padding="3dp"
          android:cacheColorHint="#dddeeeff"
          android:isScrollContainer="true"
         android:id="@+id/lvmain_list"/>
</LinearLayout>
```

文件 MainActivity.java 的功能是载入主界面的布局文件，通过方法 getDBFilesFromQQPath()从 QQ 的数据库目录复制数据库，然后用方法 setControls()动态创建按钮列表显示出已登录过的 QQ 号。MainActivity.java 文件的具体实现代码如下。

```
public class MainActivity extends Activity implements OnClickListener {
	/** Called when the activity is first created */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main_layout);

		getDBFilesFromQQPath();		//获取 DB 文件
		setControls();				//动态设置按钮
	}

	 private void getDBFilesFromQQPath()
	 {
		  try{
		  DBFiles.cleanDBFiles();
		  }catch(Exception e){
		  e.printStackTrace();
		  }
		if(!RootUtils.hasRootPermission()){
		  Toast.makeText(this, "检测到手机无法获取 ROOT 权限，程序即将退出",
					Toast.LENGTH_LONG).show();
		  this.finish();
		}
		RootUtils.RootCommand("killall com.tencent.mobileqq\n");	//启动前关掉 QQ 进程

		String str1 = "";
		String str2 = "";
		int i = Build.VERSION.SDK_INT;
		if(i >= 14){
		   str1 = System.getenv("LD_LIBRARY_PATH");
		}
		     if ((i >= 14) && (str1 != ""))
		   {
		   str2 = " env LD_LIBRARY_PATH=" + str1 + " ";
		   }
	 StringBuilder sb = new StringBuilder(str2).append("dalvikvm -cp ");
	 sb.append(getApplication().getPackageCodePath());
	 sb.append(" com.feicong.qqmsglook.DBFiles\n");
		Toast.makeText(this, "正在获取已登录 QQ 的聊天记录信息",
			   Toast.LENGTH_SHORT).show();
		RootUtils.RootCommand(sb.toString()); //执行 DBFiles 类，作用是复制 DB 文件到本程序 databases 目录

		List<String> strings = DBFiles.getDBPaths();
		Log.i("DBFiles", strings.toString());
```

```
            for (String str : strings){
              Log.i("DBFiles", str);
              if (str.length() == 0) continue;
              String permission="chmod 666 "+ str + '\n';
              RootUtils.RootCommand(permission);    //加上读写权限
            }
       }

       private void setControls()
       {
            LinearLayout layout = (LinearLayout) findViewById(R.id.linearlayout_main);
            //       layout.setOrientation(LinearLayout.VERTICAL);
            //       layout.setGravity(Gravity.CENTER);

              List<String> strings = DBFiles.getDBNames();   //获取已登录过的 QQ 记录
            for (String str : strings){
              if (str.length() == 0) continue;
              Button btn = new Button(this);
              btn.setText(str);
              btn.setTextSize((float) 24.0);
              btn.setGravity(Gravity.CENTER);
              btn.setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.
WRAP_CONTENT));
              layout.addView(btn);
              btn.setOnClickListener(MainActivity.this);
            }
            setContentView(layout);
       }

       public void onClick(View v)
       {
              String qqUin = ((Button)v).getText().toString();
              Intent intent = new Intent();
              intent.putExtra("qqUin", qqUin);
              intent.setClass(MainActivity.this, QQMainListActivity.class);
              MainActivity.this.startActivity(intent);
       }
}
```

系统主界面的执行效果如图 14-10 所示。



图 14-10　系统主界面

14.5.4 实现选择界面

当在系统主界面中单击列表中的某一个 QQ 号码时，会弹出如图 14-11 所示的选择界面。



图 14-11 选择界面

选择界面的布局实现文件是 main_list_layout.xml，功能是使用 Button 控件分别显示“好友列表”和“群列表”两个按钮，具体实现代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#dddeeeff"
    android:orientation="vertical" >
     <Button
         android:id="@+id/friends"
         android:layout_width="fill_parent"
         android:layout_height="60.0dip"
         android:layout_gravity="center"
         android:gravity="center"
         android:text="@string/friends"
         android:textSize="24dp" >
     </Button>
     <Button
         android:textSize="24dp"
         android:gravity="center"
         android:layout_gravity="center_horizontal"
         android:id="@+id/troops"
         android:layout_width="fill_parent"
         android:layout_height="60.0dip"
         android:text="@string/troops" >
     </Button>
</LinearLayout>
```

文件 QQMainListActivity.java 的功能是监听用户单击按钮操作，并根据操作执行对应的事件处理程序。文件 QQMainListActivity.java 的具体实现代码如下。

```
public class QQMainListActivity extends Activity implements OnClickListener{
    /** Called when the activity is first created */
     private Button buttonFriends;
     private Button buttonTroops;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_list_layout);
```

```
            buttonFriends = (Button) findViewById(R.id.friends);
            buttonFriends.setOnClickListener(this);
            buttonTroops = (Button) findViewById(R.id.troops);
            buttonTroops.setOnClickListener(this);
      }

      public void onClick(View v)
      {
            Intent intent = new Intent();
            Button button = (Button) v;
            String qqUin = getIntent().getStringExtra("qqUin");
            intent.putExtra("qqUin", qqUin);
            switch (button.getId()) {
                  case R.id.friends:
                        intent.setClass(QQMainListActivity.this, QQFriendListActivity.class);
                        break;
                  case R.id.troops:
                        intent.setClass(QQMainListActivity.this, QQTroopListActivity.class);
                        break;
            }
            QQMainListActivity.this.startActivity(intent);
      }
}
```

14.5.5　实现好友列表界面

在系统选择界面中单击“好友列表”按钮，将出现如图 14-12 所示的好友列表界面。

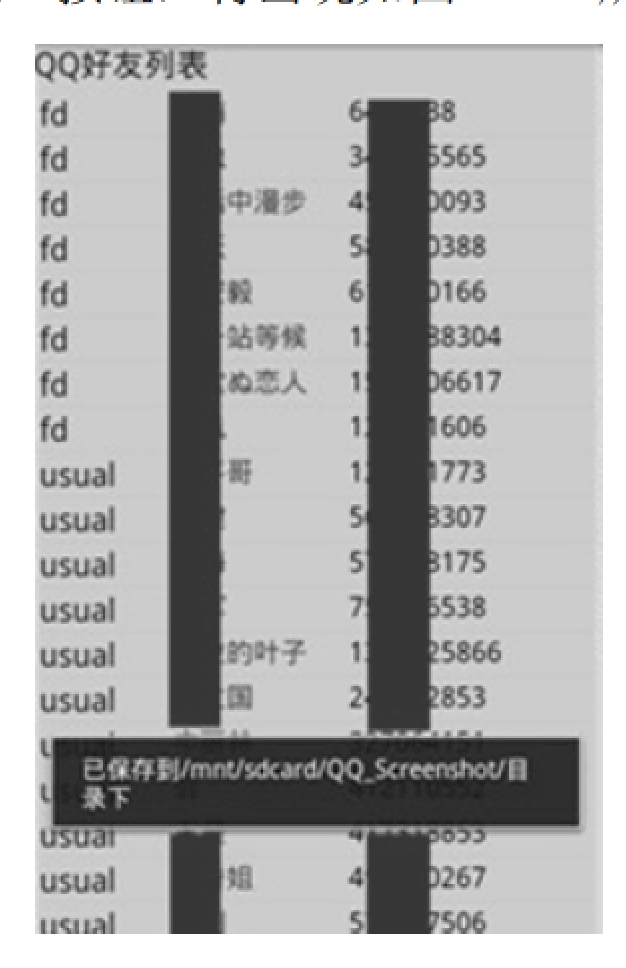


图 14-12　好友列表界面

好友列表界面的布局实现文件是 friend_list_layout.xml，功能是使用 ListView 控件列表显示被选中 QQ 的“好友列表”信息，具体实现代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
    android:background="#dddeeeff"
    android:orientation="vertical" >
     <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textColor="#0000ff"
        android:textSize="18.0dp"
        android:text="@string/qqfriend_list" >
    </TextView>
     <ListView
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:padding="3dp"
         android:cacheColorHint="#dddeeeff"
         android:isScrollContainer="true"
         android:id="@+id/lvfriend_list"/>

</LinearLayout>
```

文件 QQFriendListActivity.java 的功能是监听用户单击某个 QQ 好友的操作，并根据操作执行对应的事件处理程序。文件 QQFriendListActivity.java 的具体实现代码如下。

```
public class QQFriendListActivity extends Activity {
    /** Called when the activity is first created */
     private ListView mlistviewFriend;
     private QQFriendService mService;
     private DBHelper mDB;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.friend_list_layout);
        Intent intent = getIntent();
        String dbName = intent.getStringExtra("qqUin");
        dbName += ".db";
        mlistviewFriend = (ListView)findViewById(R.id.lvfriend_list);
        mDB = new DBHelper(this, dbName, null, 1);
        mService = new QQFriendService(mDB);
        try
          {
              SimpleAdapter adapter = new SimpleAdapter(this,
                      mService.setAdapterListData(0, -1), R.layout.friend_list,
                      new String[ ]{
                  "memberLevel", "name", "qquin"},
                  new int[ ]{
                  R.id.uin_memberLevel, R.id.uin_name, R.id.uin_number });
              mlistviewFriend.setAdapter(adapter);
          } catch (Exception e)
          {
              e.printStackTrace();
          }
          mlistviewFriend.setOnItemClickListener(new OnItemClickListener(){
```

```
                public void onItemClick(AdapterView<?> parent, View v, int location,
                        long id)
                {
                    try
                    {
                        Intent intent = new Intent();
                        TextView tv = (TextView) v
                                .findViewById((int) R.id.uin_number);
                        intent.putExtra("friendorTroopUin", tv.getText());
                        intent.putExtra("bQQ", true);
                        String qqUin = getIntent().getStringExtra("qqUin");
                        intent.putExtra("qqUin", qqUin);
                        intent.setClass(QQFriendListActivity.this,
                                QQMsgActivity.class);
                        QQFriendListActivity.this.startActivity(intent);
                    } catch (Exception e)
                    {
                        e.printStackTrace();
                    }
                }
            });
    }

    @Override
     protected void onDestroy()
     {
         mDB.close();
         super.onDestroy();
     }
}
```

14.5.6 实现聊天记录界面

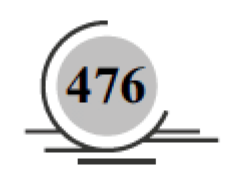
在好友列表界面中单击某个好友，将出现如图 14-13 所示的聊天记录界面。

这样距离山大啥的青岛校区都近了
【圣贤】1-糖糖(503514557) 10:32:30
学生上下学方便
【圣贤】1-糖糖(503514557) 10:32:44

【圣贤】1-糖糖(503514557) 10:32:52
我楼上是不是搞传销的啊
【太傅】8-奶油(393192879) 10:32:56
真好真好~~~终于不用慢慢悠悠的去日照威海了
【圣贤】1-糖糖(503514557) 10:32:59
一群人不停的喊
【太傅】8-奶油(393192879) 10:33:03
现在去威海都是晚上11点的车
【圣贤】1-糖糖(503514557) 10:33:04
没听明白喊什么
【圣贤】1-糖糖(503514557) 10:33:09

【太傅】8-奶油(393192879) 10:33:15
去一次熬一宿
【太傅】8-奶油(393192879) 10:33:27
疯狂英语吗，小童

图 14-13 聊天记录界面



聊天记录界面的布局实现文件是 msg_layout.xml，功能是显示被选中 QQ 和当前被选中好友的聊天记录，具体实现代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:background="#dddeeeff">

       <ListView
              android:layout_width="fill_parent"
              android:layout_height="450dp"
              android:padding="3dp"
              android:cacheColorHint="#dddeeeff"
              android:isScrollContainer="true"
              android:id="@+id/lvmsg"/>

       <RelativeLayout
                 xmlns:android="http://schemas.android.com/apk/res/android"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content">
                <Button
                      android:id="@+id/first"
                      android:layout_width="wrap_content"
                      android:layout_height="wrap_content"
                      android:layout_marginLeft="20dp"
                      android:gravity="center"
                      android:text="@string/first" />

                <Button
                     android:layout_width="wrap_content"
                     android:layout_height="wrap_content"
                     android:layout_alignTop="@id/first"
                     android:layout_toRightOf="@id/first"
                     android:layout_marginLeft="5dp"
                     android:text="@string/prev"
                     android:id="@+id/prev"
              />
                <Button
                     android:layout_width="wrap_content"
                     android:layout_height="wrap_content"
                     android:layout_alignTop="@id/prev"
                     android:layout_toRightOf="@id/prev"
                     android:layout_marginLeft="5dp"
                     android:text="@string/next"
                     android:id="@+id/next"
              />
                <Button
                     android:layout_width="wrap_content"
                     android:layout_height="wrap_content"
```

```
                    android:layout_alignTop="@id/next"
                    android:layout_toRightOf="@id/next"
                    android:layout_marginLeft="5dp"
                    android:text="@string/end"
                    android:id="@+id/end"
            />
              <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_alignBottom="@id/end"
                    android:layout_toRightOf="@id/end"
                    android:gravity="center_horizontal"
                    android:layout_marginLeft="5dp"
                    android:id="@+id/pageInfo"
                    android:textColor="#000000"
            />
        </RelativeLayout>
</LinearLayout>
```

文件 QQMsgActivity.java 的功能是监听用户单击 QQ 好友列表操作，查询记录数据库中和此好友的聊天记录。文件 QQMsgActivity.java 的具体实现代码如下。

```
public class QQMsgActivity extends Activity implements OnClickListener{
      /** Called when the activity is first created */
       private DBHelper mDB;
       private QQMessageService mService;
       private ListView listviewmsg;
       private Button buttonFirst;
       private Button buttonPrev;
       private Button buttonNext;
       private Button buttonEnd;
       private TextView pageInfo;
       private PageUtil pageUtil;

      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.msg_layout);
            listviewmsg = (ListView)findViewById(R.id.lvmsg);
            try      {
                  Intent intent = getIntent();
                  String qqUin = intent.getStringExtra("qqUin");
                  String dbName =qqUin + ".db";
                  String friendroTroopUin = intent.getStringExtra("friendorTroopUin");
                  Boolean bQQ = intent.getBooleanExtra("bQQ", true);
                  mDB = new DBHelper(this, dbName, null, 1);
                  mService = new QQMessageService(mDB, friendroTroopUin, bQQ);
                  pageUtil = new PageUtil(mService);
                  SimpleAdapter adapter = new SimpleAdapter(this,
                                    mService.setAdapterListData(0, PageUtil.pagesize),
                                    R.layout.msg, new String[ ]
                                    { "uin", "msg" }, new int[ ]
```

```
                    { R.id.uin, R.id.uin_msg });
            listviewmsg.setAdapter(adapter);
        } catch (Exception e){
            e.printStackTrace();
        }
        buttonFirst = (Button) findViewById(R.id.first);
        buttonFirst.setOnClickListener(this);
        buttonPrev = (Button) findViewById(R.id.prev);
        buttonPrev.setOnClickListener(this);
        buttonPrev.setEnabled(false);
        buttonNext = (Button) findViewById(R.id.next);
        buttonNext.setOnClickListener(this);
        buttonEnd = (Button) findViewById(R.id.end);
        buttonEnd.setOnClickListener(this);
        pageInfo = (TextView) findViewById(R.id.pageInfo);
        int index = 1;
        try    {
          int pageCount = pageUtil.getPagecount();
          if ((pageCount == 0) || (pageCount == 1)){
            buttonFirst.setEnabled(false);
            buttonPrev.setEnabled(false);
                buttonNext.setEnabled(false);
                buttonEnd.setEnabled(false);
                if (pageCount == 0) {
                index = 0;
                }
        }
            pageInfo.setText("(" + String.valueOf(index) + "/" + pageCount + ")");
        } catch (Exception e){
            buttonFirst.setEnabled(false);
            buttonPrev.setEnabled(false);
            buttonNext.setEnabled(false);
            buttonEnd.setEnabled(false);
            pageInfo.setText("(0/0)");
        }
    }

    @Override
    protected void onDestroy(){
        mDB.close();
        super.onDestroy();
    }

    public void onClick(View v){
        Button button = (Button) v;
        SimpleAdapter adapter = null;
        switch (button.getId()) {
            case R.id.first:
                pageUtil.firstPage();
                break;
            case R.id.prev:
```

```
                    pageUtil.prevPage();
                    break;
                case R.id.next:
                    pageUtil.nextPage();
                    break;
                case R.id.end:
                    pageUtil.endPage();
                    break;
            }
            try
            {
                adapter = new SimpleAdapter(this,
                        pageUtil.getList(),
                        R.layout.msg,
                        new String[ ]{ "uin", "msg" },
                        new int[ ]{ R.id.uin, R.id.uin_msg });
                listviewmsg.setAdapter(adapter);
                pageInfo.setText("(" + pageUtil.getCurrentpage() + "/" + pageUtil.getPagecount() + ")");
                if (pageUtil.getCurrentpage() == 1 && pageUtil.getPagecount() != 0) { //首页
                    buttonPrev.setEnabled(false);
                    buttonNext.setEnabled(true);
                } else if (pageUtil.getCurrentpage() == pageUtil.getPagecount()
                        && pageUtil.getPagecount() != 0) {    //尾页
                    buttonNext.setEnabled(false);
                    buttonPrev.setEnabled(true);
                } else if ((pageUtil.getPagecount() == 0) || (pageUtil.getPagecount() == 1)) {
                    buttonFirst.setEnabled(false);
                    buttonPrev.setEnabled(false);
                    buttonNext.setEnabled(false);
                    buttonEnd.setEnabled(false);
                } else {
                    buttonPrev.setEnabled(true);
                    buttonNext.setEnabled(true);
                }
            } catch (Exception e){
                e.printStackTrace();
            }
        }
}
```

到此为止，本系统实例的主要内容全部介绍完毕。有关获取 QQ 群聊天记录的方式和上述过程类似，为节省本书篇幅在此不再介绍。

注意： 本书涉及了很多Android安全方面的内容，建议读者参考本书同系列丛书中的《Android系统安全和反编译实战》一书的内容。

第 15 章　吃货选择器

经过本书前面内容的学习，相信读者已经掌握了 jQuery Mobile 移动 Web 开发技术的基本知识。在本章的内容中，将综合运用本书前面所学的知识，结合使用 HTML 5、CSS 3 和 jQuery Mobile 技术开发一个能够在移动设备中运行的吃货选择器系统。希望读者认真阅读本章内容，仔细体会 HTML 5+jQuery Mobile+CSS 组合在移动 Web 开发领域的精髓。

15.1　需 求 分 析

知识点讲解：光盘:视频\视频讲解\第 15 章\需求分析.avi

在开发一个 Web 项目之前，一定要做好需求分析工作，这是设计并开发任何软件项目的准备工作。本节将详细讲解本系统需求分析工作，为读者学习本书后面的知识打下基础。

15.1.1　背景分析

吃货，多指爱吃的人，这个词语属于中性词，目前最广泛也是大家最认可的解释是特别会吃、爱吃的人，正常环境下使用略带褒义（具体视语境而定）。吃货喜欢吃各类美食，并对美食有一种独特的向往、追求，实质上是大部分人好吃的借口罢了。

都说民以食为天，一日三餐必不可少，并且生活中的零食、下午茶和饭后甜点总是引人入胜。但是随着社会生活节奏的加快，就餐的时间越来越少。在快生活节奏的社会背景下，快速点餐成为生活中的一大需求。在现实生活中，无论是到食堂用餐，还是去餐馆就餐，在途中和排队上会浪费很多时间，并且去晚了经常会吃不到想吃的食物。为此很多人对食堂和餐馆的满意度不高，所以很多人宁愿吃已经变凉的盒饭。正因为如此，食堂和餐馆因为无法准确预测客户的需求，经常会出现有些食物因为没有卖出去只好倒掉，而用户需要的一些食物却已卖完的现象。

在上述现实背景下，吃货选择器系统便应运而生。

15.1.2　系统目标

根据前面的背景分析，本系统的目标如下。

（1）提高用户就餐的效率，节省就餐时间。

（2）用户可以提前预订，这样能够保证可以品尝到自己喜欢的菜肴。

（3）餐馆和饭店根据用户的预定需求，提前准备好饭菜，避免不必要的浪费。

（4）订餐方式灵活方便，只需在手机或平板电脑等移动设备中轻轻点几下，即可完成订餐操作。

15.1.3　系统模块划分

根据前面背景分析和系统目标介绍，本章吃货点餐系统需要具备如下所示的模块。

（1）面向全国，可以选择城市

本系统具有良好的可扩展性，不仅仅局限于对某一个区域的用户服务，而且可以面向全国，甚至全球。用户进入系统后，可以选择就餐城市。

（2）选择菜品和食物

为了满足用户的不同口味及对菜品的不同需求，系统推出了各种各样的食物和菜品供用户选择，这样的服务更能使客户满意。

（3）多家商家供用户选择

在用户选择的每座城市中，提供多家商家，并且经过选择食物和城市步骤之后，系统能够直接显现用户喜爱的餐馆，这样大大提高了点餐效率，使整个服务更加人性化。

（4）商家介绍

在系统中提供商家完善的信息，包括地址、特色菜品、电话、地图位置等信息，这些信息都是就餐用户最为关注的。

（5）用户评价

为了提高系统信息的真实性，让用户完成一次完美的就餐体验，系统特意推出用户评价模块，就餐完毕，用户可以对商家进行点评。这些点评信息对以后就餐的用户来说，能够起到很好的参考作用。

综上所述，本章吃货选择器系统的模块结构如图 15-1 所示。

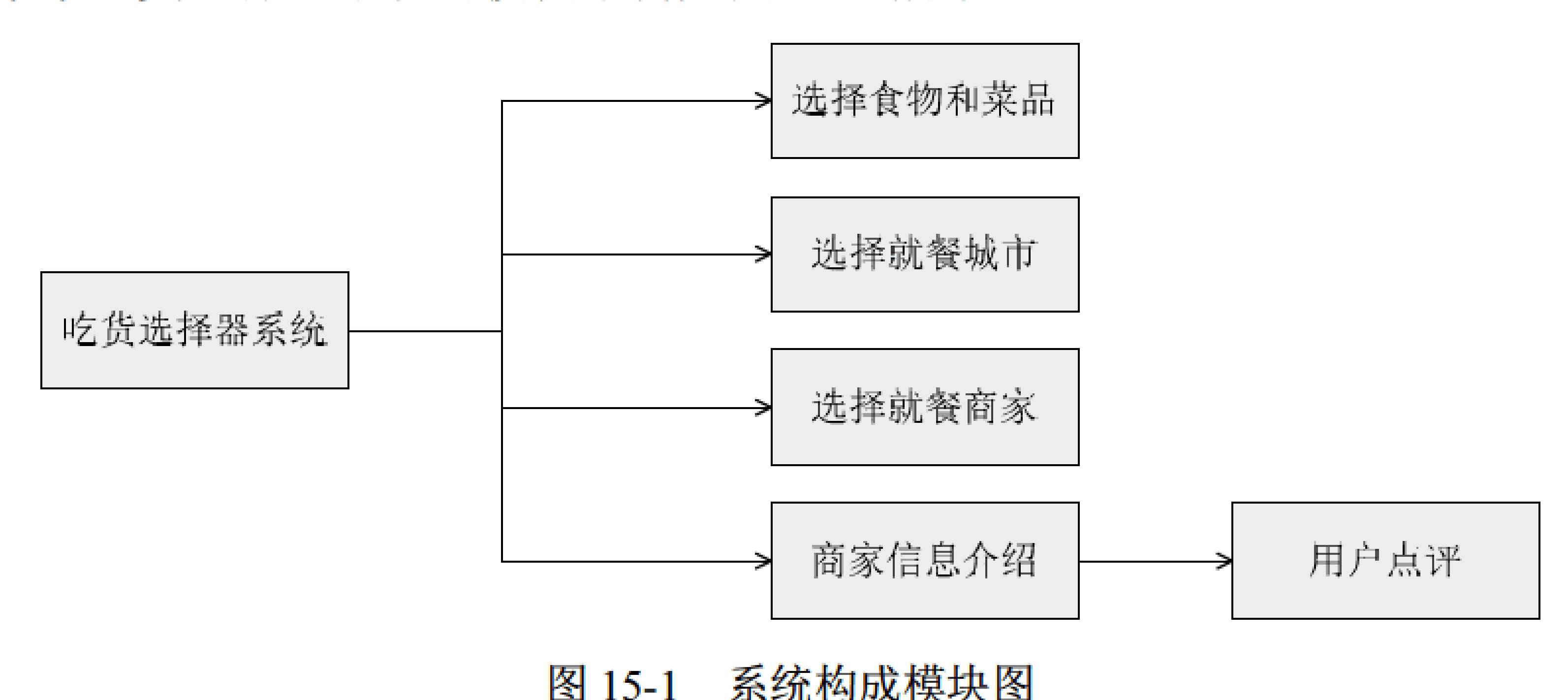


图 15-1　系统构成模块图

15.2　界面设计

知识点讲解：光盘:视频\视频讲解\第 15 章\界面设计.avi

根据 15.1 节的需求分析可知需要设计多个界面。根据移动设备的尺寸要求，特别是手机屏幕的尺寸需求，进行如下界面设计。

（1）在主界面显示选择食物和菜品菜单，提供给用户不同板块之间的选择，每一个选择都带有一个图像说明板，单击每一个选项后将进入第 2 个页面。首页设计草图效果如图 15-2 所示。

（2）在第 2 个页面中，用户可以选择就餐城市。城市以列表形式显示，在城市后面会显示该城市的餐馆数目信息，单击每个城市选项后会进入第 3 个页面。第 2 个页面设计草图效果如图 15-3 所示。

（3）在第 3 个页面中显示某个城市的餐馆信息，以列表的样式逐一显示每一家餐馆的图片、名字以及评价信息。在标题栏中显示应用程序的 Logo 和“后退”按钮，用户可以单击返回上一步。单击列表中的某一个餐厅后可以来到第 4 个页面。第 3 个页面设计草图效果如图 15-4 所示。



图 15-2　系统主页的设计草图

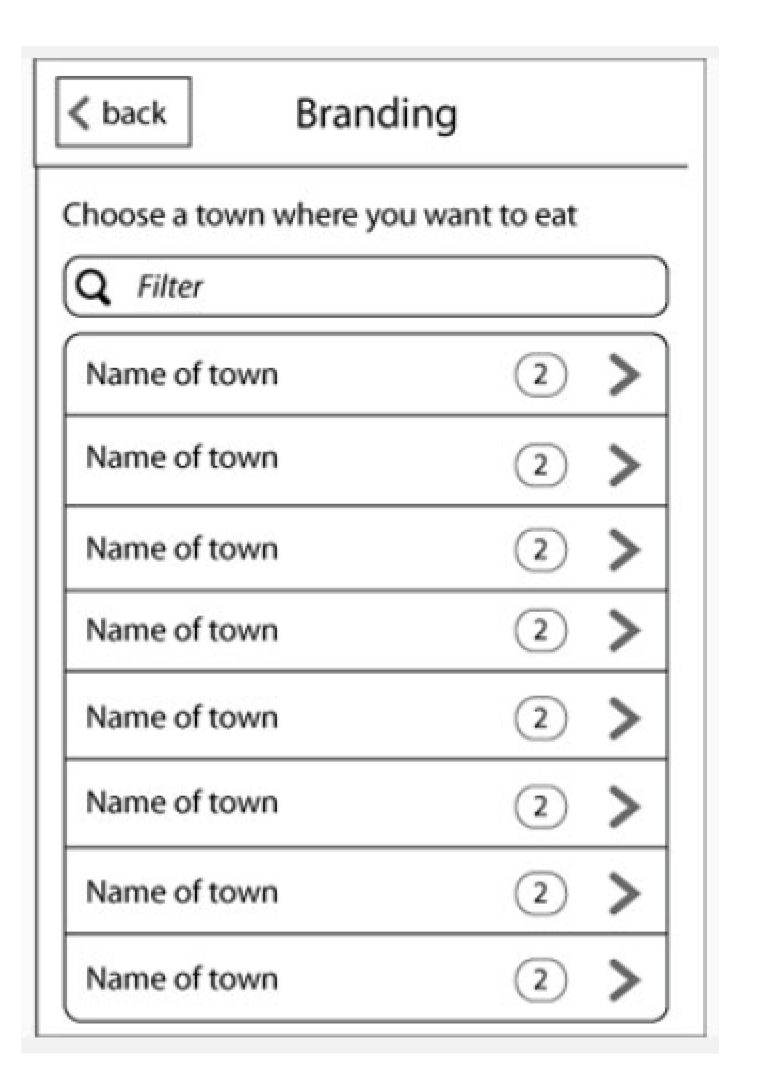


图 15-3　第 2 个页面的设计草图

（4）在第 4 个页面中显示某个餐馆的详细信息，包括名称、特色、地址、电话、地图定位和用户评价等信息。通过调用谷歌地图的方式，可以在地图中定位显示此餐馆的位置。只需一个链接就可以让用户打开谷歌地图（无论是使用浏览器或谷歌地图应用程序，这都取决于当前设备的支持状况），并在地图上找到餐馆。第 4 个页面设计草图效果如图 15-5 所示。

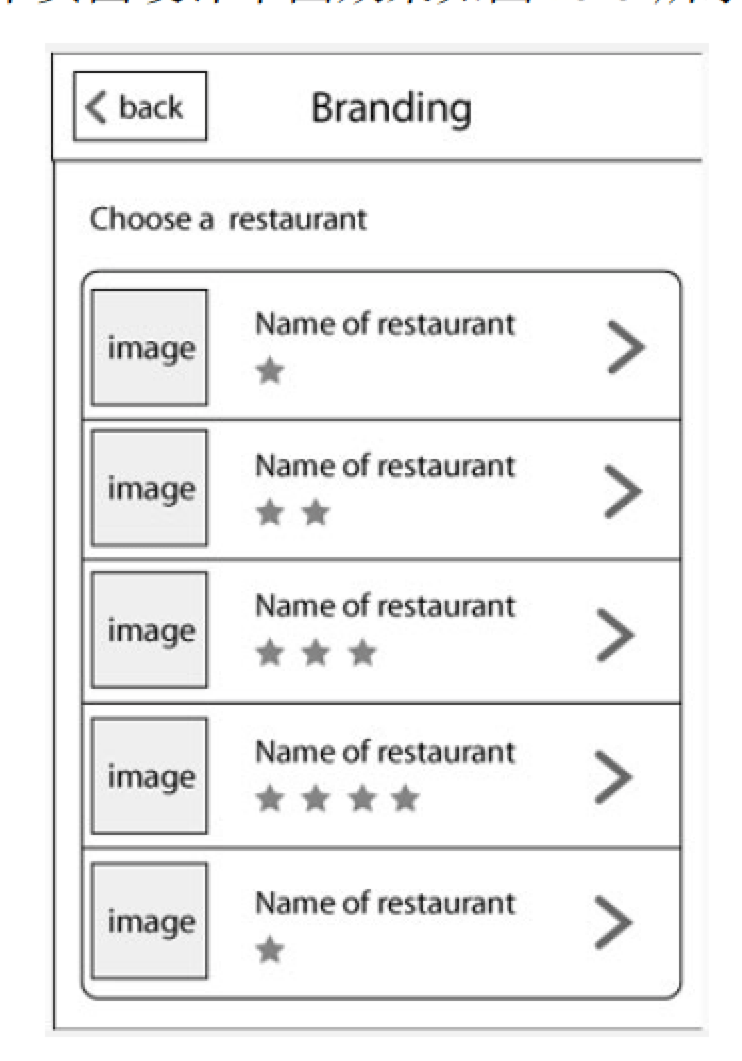


图 15-4　第 3 个页面的设计草图

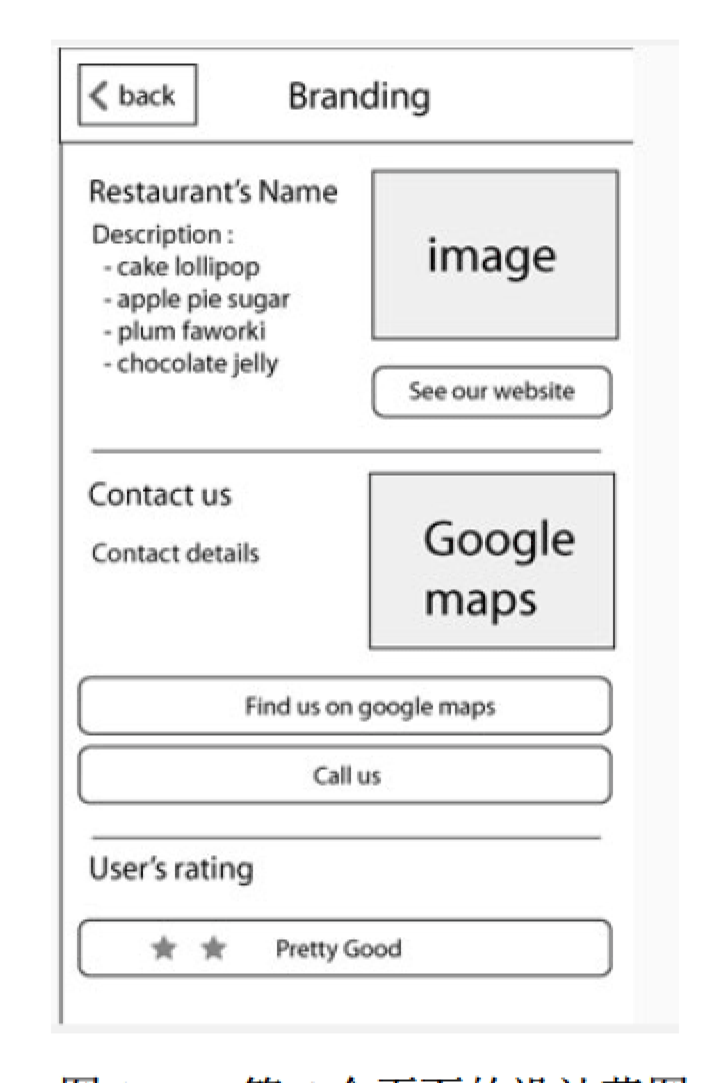


图 15-5　第 4 个页面的设计草图

15.3　构建 jQuery Mobile 平台

知识点讲解：光盘:视频\视频讲解\第 15 章\构建 jQuery Mobile 平台.avi

本系统需要借助于 jQuery Mobile 技术实现，所以需要在页面中事先构建 HTML+jQuery Mobile+CSS 平

台。构建工作需要在 HTML 文件的头文件中实现，具体实现代码如下所示。

```
<head>
  <meta charset="UTF-8">
     <title>点餐系统</title>
     <meta name="viewport" content="width=device-width, initial-scale=1">
     <link rel="stylesheet" href="jquery.mobile.structure-1.0.1.css" />
     <link rel="apple-touch-icon" href="images/launch_icon_57.png" />
     <link rel="apple-touch-icon" sizes="72x72" href="images/launch_icon_72.png" />
     <link rel="apple-touch-icon" sizes="114x114" href="images/launch_icon_114.png" />
     <link rel="stylesheet" href="jquery.mobile-1.0.1.css" />
     <link rel="stylesheet" href="custom.css" />
     <script src="js/jquery-1.7.1.min.js"></script>
     <script src="js/jquery.mobile-1.0.1.min.js"></script>
</head>
```

15.4 页 面 实 现

知识点讲解：光盘:视频\视频讲解\第 15 章\页面实现.avi

经过系统分析、模块划分、界面设计和平台搭建工作之后，接下来正式步入系统实现阶段的工作。在本节将详细讲解 HTML 5 页面的具体实现过程。

15.4.1 第 1 个页面——系统主页

本实例的系统主页文件是 index.html，在顶部显示系统 Logo 图标，下方列表显示各种美味产品系列。在每一个列表选项中显示产品的图片、名称和到第 2 个页面的链接图标。文件 index.html 的具体实现代码如下。

```
<!DOCTYPE html>
<html>
     <head>
     <meta charset="UTF-8">
     <title>点餐系统</title>

     <meta name="viewport" content="width=device-width, initial-scale=1">
     <link rel="stylesheet" href="jquery.mobile.structure-1.0.1.css" />
     <link rel="apple-touch-icon" href="images/launch_icon_57.png" />
     <link rel="apple-touch-icon" sizes="72x72" href="images/launch_icon_72.png" />
     <link rel="apple-touch-icon" sizes="114x114" href="images/launch_icon_114.png" />
     <link rel="stylesheet" href="jquery.mobile-1.0.1.css" />
     <link rel="stylesheet" href="custom.css" />
     <script src="js/jquery-1.7.1.min.js"></script>
     <script src="js/jquery.mobile-1.0.1.min.js"></script>
     </head>

<body>
<div data-role="page" id="home" data-theme="c">

     <div data-role="content">
```

```
        <div id="branding">
            <h1>快乐点餐系统</h1>
        </div>

        <div class="choice_list">
        <h1> 亲，您想品尝什么美味？</h1>

        <ul data-role="listview" data-inset="true" >
        <li><a href="choose_town.html" data-transition="slidedown"> <img src="sushis.jpg"/> <h3> 来点寿司
</h3></a></li>
        <li><a href="choose_town.html"  data-transition="slidedown"> <img src="pizza.jpg"/> <h3> 一个披萨
</h3></a></li>
        <li><a href="choose_town.html"  data-transition="slidedown"> <img src="kebap.jpg"/> <h3> 大块烤肉
</h3></a></li>
        <li><a href="choose_town.html"  data-transition="slidedown"> <img src="burger.jpg"/> <h3> 来个汉堡
</h3></a></li>
        <li><a href="choose_town.html"  data-transition="slidedown"> <img src="nems.jpg"/> <h3> 素味小吃
</h3></a></li>
        <li><a href="choose_town.html"  data-transition="slidedown"> <img src="tradi.jpg"/> <h3> 更多大众口味
欢迎您</h3></a></li>
        </ul>

        </div>
        </div>

</div><!-- /page -->
</body>
</html>
```

执行后的效果如图 15-6 所示。



图 15-6　第 1 个页面执行效果

15.4.2　第 2 个页面——选择城市

本系统实例的第 2 个页面是选择城市页面 choose_town.html，在页面顶部显示一个过滤文本框供用户输入城市名，在下面列表显示各个城市，在每一个城市列表选项后面显示到第 3 个页面的链接。文件 choose_town.html 的具体实现代码如下。

```
<!DOCTYPE html>
```

```
<html>
<head>
     <meta charset="UTF-8">
    <title>Restaurant Picker</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="jquery.mobile.structure-1.0.1.css" />
    <link rel="apple-touch-icon" href="images/launch_icon_57.png" />
    <link rel="apple-touch-icon" sizes="72x72" href="images/launch_icon_72.png" />
    <link rel="apple-touch-icon" sizes="114x114" href="images/launch_icon_114.png" />
    <link rel="stylesheet" href="jquery.mobile-1.0.1.css" />
    <link rel="stylesheet" href="custom.css" />
    <script src="js/jquery-1.7.1.min.js"></script>
    <script src="js/jquery.mobile-1.0.1.min.js"></script>
</head>
<body>
<div id="choisir_ville" data-role="page" data-add-back-btn="true">

    <div data-role="header">
        <h1> 点餐系统</h1>
    </div>

    <div data-role="content">

    <div class="choice_list">
    <h1> 亲爱的吃货朋友，您准备在哪座城市就餐？</h1>

    <ul data-role="listview" data-inset="true" data-filter="true"   >
    <li><a href="choose_restaurant.html"   data-transition="slidedown">北京<span class="ui-li-count" > 3
</span></a> </li>
    <li><a href="choose_restaurant.html"   data-transition="slidedown">上海<span class="ui-li-count" > 2
</span></a> </li>
    <li><a href="choose_restaurant.html" data-transition="slidedown">广州<span class="ui-li-count" > 5
</span></a> </li>
    <li><a href="choose_restaurant.html" data-transition="slidedown">深圳<span class="ui-li-count" > 1
</span></a> </li>
    <li><a href="choose_restaurant.html" data-transition="slidedown">济南<span class="ui-li-count" > 2
</span></a> </li>
    <li><a href="choose_restaurant.html" data-transition="slidedown">成都<span class="ui-li-count" > 2
</span></a> </li>
    <li><a href="choose_restaurant.html" data-transition="slidedown">武汉<span class="ui-li-count" > 10
</span></a> </li>
    <li><a href="choose_restaurant.html" data-transition="slidedown">昆明<span class="ui-li-count" > 8
</span></a> </li>
    <li><a href="choose_restaurant.html" data-transition="slidedown">长沙<span class="ui-li-count" > 1
</span></a> </li>
    <li><a href="choose_restaurant.html" data-transition="slidedown">青岛<span class="ui-li-count" > 3
</span></a> </li>
    <li><a href="choose_restaurant.html" data-transition="slidedown">大连<span class="ui-li-count" > 2
</span></a> </li>
    <li><a href="choose_restaurant.html" data-transition="slidedown">沈阳<span class="ui-li-count" > 4
</span></a> </li>
```

```
        <li><a href="choose_restaurant.html" data-transition="slidedown">兰州<span class="ui-li-count" > 6
</span></a> </li>
        <li><a href="choose_restaurant.html" data-transition="slidedown">拉萨<span class="ui-li-count" > 2
</span></a> </li>
        <li><a href="choose_restaurant.html" data-transition="slidedown">重庆<span class="ui-li-count" > 1
</span></a> </li>
        <li><a href="choose_restaurant.html" data-transition="slidedown">天津<span class="ui-li-count" > 5
</span></a> </li>
        <li><a href="choose_restaurant.html" data-transition="slidedown">开封<span class="ui-li-count" > 1
</span></a> </li>
        <li><a href="choose_restaurant.html" data-transition="slidedown">西安<span class="ui-li-count" > 3
</span></a> </li>
        <li><a href="choose_restaurant.html" data-transition="slidedown">日照<span class="ui-li-count" > 8
</span></a> </li>
        <li><a href="choose_restaurant.html" data-transition="slidedown">南京<span class="ui-li-count" > 5
</span></a> </li>
        <li><a href="choose_restaurant.html" data-transition="slidedown">杭州<span class="ui-li-count" > 3
</span></a> </li>
        <li><a href="choose_restaurant.html" data-transition="slidedown">苏州<span class="ui-li-count" > 5
</span></a> </li>
        <li><a href="choose_restaurant.html" data-transition="slidedown">莒县<span class="ui-li-count" > 2
</span></a> </li>
        </ul>
        </div>

        </div>

</div><!-- /page -->
</body>
</html>
```

执行后的效果如图 15-7 所示。



图 15-7 第 2 个页面执行效果

15.4.3 第 3 个页面——商家列表

本系统的第 3 个页面是商家列表页面 choose_restaurant.html，列表显示某个城市的餐馆信息，在每一个餐馆列表选项中显示餐馆图片、餐馆名和评价信息，并且在最后显示到第 4 个页面的链接。文件 choose_restaurant.html 的具体实现代码如下。

```
<!DOCTYPE html>
<html>
<head>
	<meta charset="UTF-8">
	<title>Restaurant Picker</title>
	<meta name="viewport" content="width=device-width, initial-scale=1">
	<link rel="stylesheet" href="jquery.mobile.structure-1.0.1.css" />
	<link rel="apple-touch-icon" href="images/launch_icon_57.png" />
	<link rel="apple-touch-icon" sizes="72x72" href="images/launch_icon_72.png" />
	<link rel="apple-touch-icon" sizes="114x114" href="images/launch_icon_114.png" />
	<link rel="stylesheet" href="jquery.mobile-1.0.1.css" />
	<link rel="stylesheet" href="custom.css" />
	<script src="js/jquery-1.7.1.min.js"></script>
	<script src="js/jquery.mobile-1.0.1.min.js"></script>
</head>
<body>
<div id="choisir_restau" data-role="page" data-add-back-btn="true">

	<div data-role="header">
		<h1> 订餐系统</h1>
	</div>

	<div data-role="content">

	<div class="choice_list">
	<h1> 选择一家餐厅吧！</h1>

	<ul data-role="listview" data-inset="true" >
	<li><a href="restaurant.html" data-transition="slidedown"> <img src="restau01_mini.jpg"/> <h2> 四季鱼香
</h2> <p class="classement four"> 4 stars    </p></a></li>
	<li><a href="restaurant.html" data-transition="slidedown"> <img src="restau02_mini.jpg "/> <h2> 风花雪月夜
</h2> <p class="classement four"> 4 stars    </p> </a></li>
	<li><a href="restaurant.html" data-transition="slidedown"> <img src="restau03_mini.jpg "/> <h2> 赛江南
</h2> <p class="classement one"> 1 star    </p> </a></li>
	<li><a href="restaurant.html" data-transition="slidedown"> <img src="restau04_mini.jpg "/> <h2> 塞外大烤肉
</h2> <p class="classement three"> 3 stars    </p> </a></li>
	<li><a href="restaurant.html" data-transition="slidedown"> <img src=" restau05_mini.jpg"/> <h2> 两岸咖啡
</h2> <p class="classement two"> 2 stars </p> </a></li>
	</ul>

	</div>
	</div>

</div><!-- /page -->
</body>
</html>
```

执行后的效果如图 15-8 所示。



图 15-8　第 3 个页面执行效果

15.4.4　第 4 个页面——商家详情

本系统实例的第 4 个页面是商家详情页面 restaurant.html，功能是详细显示这家餐馆的信息。在顶部显示餐馆的名称、图片、评价信息、特色菜品、地址和联系电话等信息。文件 restaurant.html 的具体实现代码如下。

```
<!DOCTYPE html>
<html>
<head>
	<meta charset="UTF-8">
	<title>Restaurant Picker</title>
	<meta name="viewport" content="width=device-width, initial-scale=1">
	<link rel="stylesheet" href="jquery.mobile.structure-1.0.1.css" />
	<link rel="apple-touch-icon" href="images/launch_icon_57.png" />
	<link rel="apple-touch-icon" sizes="72x72" href="images/launch_icon_72.png" />
	<link rel="apple-touch-icon" sizes="114x114" href="images/launch_icon_114.png" />
	<link rel="stylesheet" href="jquery.mobile-1.0.1.css" />
	<link rel="stylesheet" href="custom.css" />
	<script src="js/jquery-1.7.1.min.js"></script>
	<script src="js/jquery.mobile-1.0.1.min.js"></script>
</head>
<body>
<div id="restau" data-role="page" data-add-back-btn="true">

	<div data-role="header">
		<h1> 点餐系统</h1>
	</div>

	<div data-role="content">
	<div class="ui-grid-a" id="restau_infos">
		<div class="ui-block-a">
```

```
            <h1> 四季渔乡</h1>
            <p><strong>   什刹海畔的一家情侣特色餐厅</strong></p>
            <p> 菜单招牌菜: </p>
                <ul>
                    <li> 情人脸的红</li>
                    <li> 娇娇脸儿媚 </li>
                    <li> 昨夜风雨声 </li>
                </ul>
            </div>
            <div class="ui-block-b">
            <p><img src="restau01.jpg" alt="jeannette photo"/></p>
            <p><a href="http://www.toppr.net" rel="external" data-role="button"> 登录四季渔乡的官网</a></p>
            </div>
        </div><!-- /grid-a -->
        <hr/>

        <div class="ui-grid-a" id="contact_infos">
            <div class="ui-block-a">
            <h2> 联系我们</h2>
            <p>什刹海桃花岛 2 弄</p>
            <p> 西单分店，西直门 27 号 </p>
            </div>
            <div class="ui-block-b">
            <img src="01_maps.jpg" alt="plan jeanette"/>
            </div>
        </div><!-- /grid-a -->
        <div id="contact_buttons">
            <a
href="http://maps.google.fr/maps?q=jeannette+et+les+cycleux&hl=fr&sll=46.75984,1.738281&sspn=10.221882,
15.764648&hq=jeannette+et+les+cycleux&t=m&z=13&iwloc=A" data-role="button" data-icon="maps"> 在地图
定位 </a>
            <a href="tel:0388161072"   data-role="button" data-icon="tel"> 电话我们吧 </a>
        </div>
        <hr/>

        <div id="notation">
        <form>
        <label for="select-choice-0" class="select"><h2> 用户评价 </h2></label>
            <select name="note_utilisateur" id="note_utilisateur" data-native-menu="false" data-theme="c" >
                <option value="one" class="one"> 不参与了，谢谢！</option>
                <option value="two" class="two">一般般吧 </option>
                <option value="three" class="three">好极了 </option>
                <option value="four" class="four">极力推荐！</option>
            </select>
        </form>
        </div>


        <script type="text/javascript">

        $( '#restau' ).live( 'pageinit',function(event){
```

```
            var SelectedOptionClass = $('option:selected').attr('class');
            $('div.ui-select').addClass(SelectedOptionClass);

            $('#note_utilisateur').live('change', function(){
                $('div.ui-select').removeClass(SelectedOptionClass);

                SelectedOptionClass = $('option:selected').attr('class');
                $('div.ui-select').addClass(SelectedOptionClass);

             });


        });

        </script>


        </div>
</div><!-- /page -->
</body>
</html>
```

在上述代码中实现了两个按钮："在地图定位"和"电话我们吧"。如果在真实手机中执行，单击"在地图定位"按钮后会询问用户是否要使用浏览器打开，单击"是"按钮则会调用手机内安装的谷歌地图应用程序进行定位。如果单击"电话我们吧"按钮，则直接进入拨号界面拨打电话。在计算机中运行时，单击"电话我们吧"按钮后会弹出"浏览器不支持"的提示信息。

执行后的效果如图 15-9 所示。



图 15-9　第 4 个页面执行效果

15.5　编写样式文件

知识点讲解：光盘:视频\视频讲解\第 15 章\编写样式文件.avi

编写 HTML 5 页面文件之后，本节将详细讲解 CSS 样式修饰文件 custom.css 的具体实现过程，为读者学习本书后面的知识打下基础。

15.5.1 设置基本样式

首先设置 4 个 HTML 5 页面文件的基本样式，包括字体、背景图片、背景颜色、图像和列表等元素，具体实现代码如下。

```
/*** general styling */
.ui-page.ui-body-c{
background:url(images/bg.png);
box-shadow:   0px 0px 30px 5px rgba(107, 105, 105, 0.3) inset,
 0px 0px 0px 1px rgba(107, 105, 105, 0.4) inset;

}

.ui-icon.ui-icon-arrow-r {
background-color:rgb(136, 111, 110);
}
.ui-corner-all,
.ui-corner-top,
.ui-corner-bottom,
.ui-corner-tl,
.ui-corner-tr,
.ui-corner-bl,
.ui-header .ui-btn-corner-all,
.ui-listview-filter .ui-btn-corner-all,
#restau_infos .ui-btn-corner-all,
#contact_buttons .ui-btn-corner-all,
#notation .ui-btn-corner-all{
border-radius:0.2em;
}

.ui-btn-active {
background: #654644; /* Old browsers */
background: -moz-linear-gradient(top,   #654644 0%, #331c1b 100%); /* FF3.6+ */
background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#654644), color-stop(100%,#331c1b));
/* Chrome,Safari4+ */
background: -webkit-linear-gradient(top,   #654644 0%,#331c1b 100%); /* Chrome10+,Safari5.1+ */
background: -o-linear-gradient(top,   #654644 0%,#331c1b 100%); /* Opera 11.10+ */
background: -ms-linear-gradient(top,   #654644 0%,#331c1b 100%); /* IE10+ */
background: linear-gradient(top,   #654644 0%,#331c1b 100%); /* W3C */
color:#fff !important;
}
.ui-content .choice_list   .ui-btn-active .ui-link-inherit,
.ui-btn-down-c a.ui-link-inherit,
#home .ui-btn-down-c a.ui-link-inherit{
color:#fff !important;
}

img{
max-width: 100%;
height: auto; width: auto;
-webkit-box-sizing: border-box;
```

```
-moz-box-sizing: border-box;
box-sizing: border-box;
}
.ui-grid-a .ui-block-a, .ui-grid-a .ui-block-b {
	width: 48%;
	padding:1%;
}
```

15.5.2　设置标题栏的样式

在前面的 4 个 HTML 5 文件中都有标题栏，设置标题栏样式的代码如下。

```
.ui-header.ui-bar-a{
background:url(images/header_bg.png);
}
.ui-header .ui-title {
text-indent:-9999px;
font-size:0px;
background:url(images/header_logo.png) no-repeat 69% 5px ;
height:33px;
padding:5px 0 5px 50px;
margin:0px;
}

.ui-header  .ui-btn-up-a  {
background:rgba(255, 255, 255, 0.1);
box-shadow:none;
}
.ui-header  .ui-btn-hover-a {
background:rgba(0, 0, 0, 0.3);
box-shadow:none;
}
```

15.5.3　设置系统主页的样式

本系统的主页是 index.html，修饰此页面的样式代码如下。

```
#branding{
background:url(images/logo.png) no-repeat;
width:322px;
height:165px;
text-indent:-999px;
font-size:0px;
margin:-10px auto 0 auto;
border-bottom:1px solid rgba(65, 38, 37, 0.6);
}

.choice_list h1{
margin-top:30px;
font-size:18px;
color:rgb(65, 38, 37);
```

```
font-weight:normal;
font-style:italic;
padding:5px 0 6px 50px;
background:url(images/pagination.png) no-repeat;
}

#home .choice_list h1{
background-position: 0 -16px;
}
#home .choice_list   h3{
padding-top:10px;
color:rgb(63, 41, 39);
}
#home .choice_list .ui-btn-active   a.ui-link-inherit h3{
color:#fff;
}
.choice_list   img{
padding:3px;
}
```

执行之后的效果如图 15-10 所示。



图 15-10　修饰第 1 个页面后的执行效果

15.5.4　修饰第 2 个页面

本系统的第 2 个页面是 choose_town.html，修饰此页面的 CSS 样式代码如下。

```
#choisir_ville .choice_list h1{
background-position: 0 -72px;
margin-bottom:20px;
}
#choisir_ville .choice_list a{
padding-top:10px;
color:rgb(63, 41, 39);
```

```
}
#choisir_ville .ui-listview-filter a{
padding-top:0px;
}
```

执行之后的效果如图 15-11 所示。



图 15-11　修饰第 2 个页面后的执行效果

15.5.5　修饰第 3 个页面

本系统的第 3 个页面是 choose_restaurant.html，修饰此页面的 CSS 样式代码如下。

```
#choisir_restau .choice_list h1{
background-position: 0 -132px;
margin:10px auto 20px auto;
}
#choisir_restau .choice_list a{
padding-top:10px;
color:rgb(63, 41, 39);
}

#choisir_restau .classement{
display:inline-bloc;
background:url(images/pagination.png) no-repeat   0 -182px;
height:22px;
text-indent:-999px;
font-size:0px;
}

#choisir_restau .one{
width:30px;
}
#choisir_restau .two{
width:55px;
```

```
}
#choisir_restau .three{
width:75px;
}
#choisir_restau .four{
width:99px;
}
```

然后编写修饰代码在列表中实现评价星星效果，具体实现代码如下。

```
#note_utilisateur-menu a{
padding-left:100px;
position:relative;
}

#note_utilisateur-button span.ui-btn-text{
background:url(images/pagination.png) no-repeat;
}

#note_utilisateur-button span.ui-btn-inner{
        padding-left:5px;
}

#note_utilisateur-menu li a:before{
       content: "      ";
       display:inline-block;
       width:115px;
       height:20px;
       background:url(images/pagination.png) no-repeat;
       position:absolute;
       left:0px;
}
.one #note_utilisateur-button span.ui-btn-text,
#note_utilisateur-menu li a:before{
       background-position: -75px -182px;
}
.two #note_utilisateur-button span.ui-btn-text,
#note_utilisateur-menu li + li a:before{
       background-position: -52px -182px;
}
.three #note_utilisateur-button span.ui-btn-text,
#note_utilisateur-menu li + li +li a:before{
       background-position: -27px -182px;
}
.four #note_utilisateur-button span.ui-btn-text,
#note_utilisateur-menu li + li +li +li a:before{
       background-position: -2px -182px;
}
```

执行之后的效果如图 15-12 所示。



图 15-12　修饰第 3 个页面后的执行效果

15.5.6　修饰第 4 个页面

本系统的第 4 个页面是 restaurant.html，修饰此页面的 CSS 样式代码如下。

```
#restau_infos,
#contact_infos {
color:rgb(63, 41, 39);
font-size:14px;
}
#restau_infos h1,
#contact_infos h2,
#notation h2{
color:rgb(63, 41, 39);
font-size:18px;
margin:0 auto 5px auto;
}

#restau_infos p,
#restau_infos ul,
#contact_infos p{
margin:2px auto 5px auto;
}
#restau_infos ul{
padding:0 0 0 10px;
}
#restau_infos ul   li{
list-style-type:square;
margin-left:5px;
}

#restau_infos .ui-block-b .ui-btn {
font-size:12px;
}
#restau_infos .ui-block-b .ui-btn-inner{
```

```
padding:5px;
}

#contact_buttons a{
color:rgb(63, 41, 39);
}

.ui-icon-maps {
background: rgb(63, 41, 39) url(images/maps.png) no-repeat;
}
.ui-icon-tel{
background: rgb(63, 41, 39) url(images/phone.png) no-repeat;
}
```

执行之后的效果如图 15-13 所示。



图 15-13 修饰第 4 个页面后的执行效果

单击“在地图定位”按钮后的执行效果如图 15-14 所示。

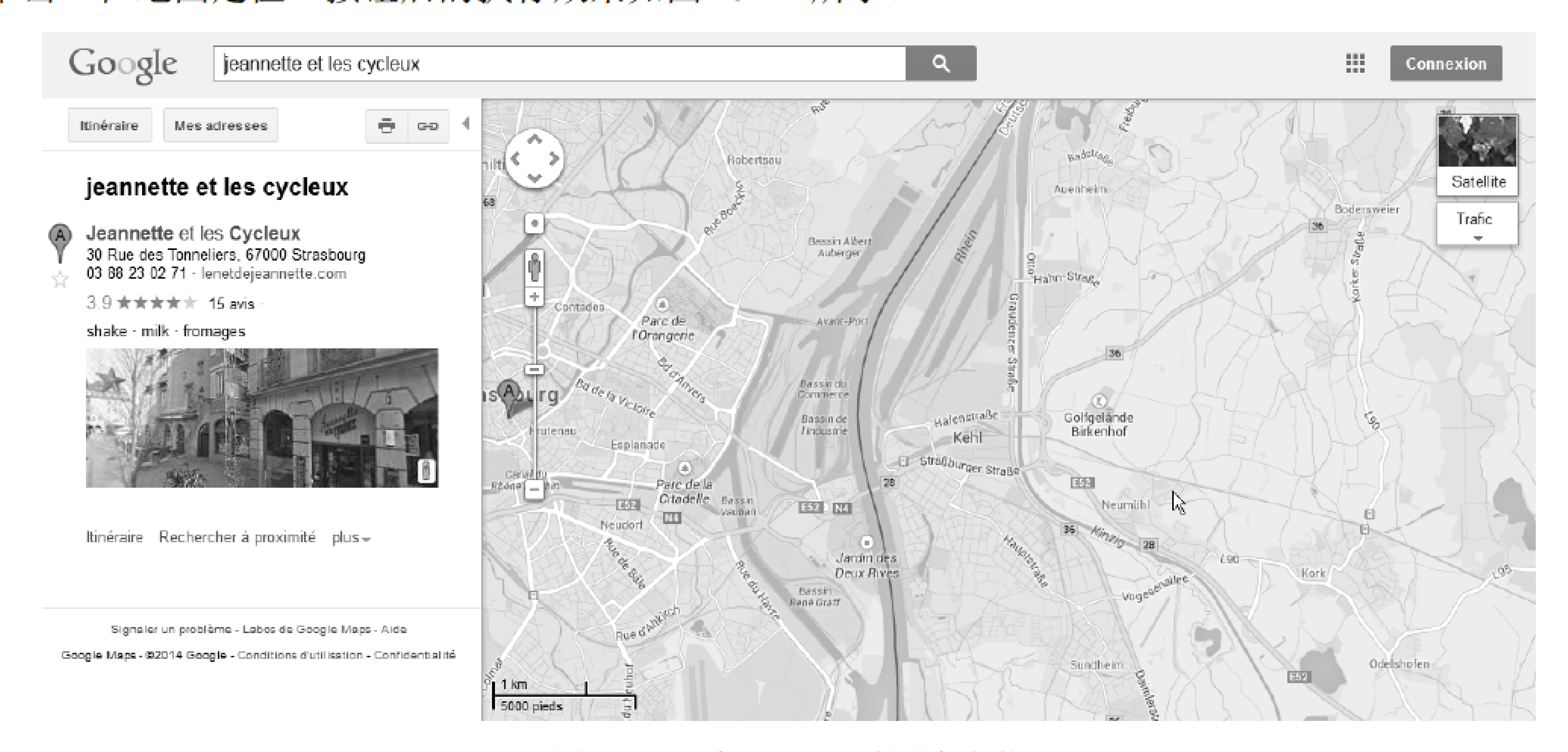


图 15-14 在 Google 地图中定位

单击“用户评价”后的按钮后，执行效果如图 15-15 所示。



图 15-15　用户评价界面效果

除了 HTML 5 和 CSS 文件之外，本系统还使用了 JavaScript 技术，特别是本书中的核心内容 jQuery Mobile。在本章 HTML 5 文件的头部文件中实现了对外部 jQuery Mobile 文件的引用，这些文件包括 jquery.mobile-1.0.1.js、jquery.mobile-1.0.1.min.js 和 jquery-1.7.1.min.js，并且引用 JavaScript 文件 addstarscript.js 实现用户评价的特效处理，具体实现代码如下。

```
$(document).ready(function() {
	var SelectedOptionClass = $('option:checked').attr('class');
	$('div.ui-select').addClass(SelectedOptionClass);

	$('#note_utilisateur').live('change', function(){
		$('div.ui-select').removeClass(SelectedOptionClass);

		SelectedOptionClass = $('option:checked').attr('class');
		$('div.ui-select').addClass(SelectedOptionClass);

	 });
});
```

到此为止，本章的“吃货选择器”系统实例的实现过程全部介绍完毕。

第 16 章　智能心率计

近几年来，各大科技巨头都纷纷推出了智能可穿戴设备。随着人们对新技术接受度的提高，可穿戴设备必将成为未来人们生活的必需品之一。开发可穿戴设备需要掌握硬件和软件技术，因为本书讲解的是 Android 可穿戴技术开发，所以本书内容将以应用软件程序开发为主。本章将详细讲解开发 Android 智能心率计的基本知识，为学习本书后面的知识打下基础。

16.1　什么是心率

知识点讲解：光盘:视频\视频讲解\第 16 章\什么是心率.avi

心率（Heart Rate）是用来描述心动周期的专业术语，是指心脏每分钟跳动的次数，以第一声音为准。现代汉语将心率解释为“心脏跳动的频率”。频率就是在单位时间内，某件事情发生的次数。两种解释合起来就是，心脏在一定时间内跳动的次数，也就是在一定时间内，心脏跳动的快慢。

据科学研究发现，正常成年人安静时的心率有显著的个体差异，平均在 75 次/分左右（60～100 次/分之间）。心率可因年龄、性别及其他生理情况而不同。初生儿的心率很快，可达 130 次/分以上。在成年人中，女性的心率一般比男性稍快。同一个人，在安静或睡眠时心率减慢，运动时或情绪激动时心率加快，在某些药物或神经体液因素的影响下，会使心率发生加快或减慢。经常进行体力劳动和体育锻炼的人，平时心率较慢。近年，国内大样本健康人群调查发现，国人男性静息心率的正常范围为 50～95 次/分，女性为 55～95 次/分，所以心率随年龄、性别和健康状况的变化而变化。

健康成人的心率为 60～100 次/分，大多数为 60～80 次/分，女性稍快；3 岁以下的幼儿常在 100 次/分以上；老年人偏慢。成人每分钟心率超过 100 次/分（一般不超过 160 次/分）或婴幼儿超过 150 次/分者，称为窦性心动过速，常见于正常人运动、兴奋、激动、吸烟、饮酒和喝浓茶后，也可见于发热、休克、贫血、甲亢、心力衰竭及应用阿托品、肾上腺素、麻黄素等。如果心率在 160～220 次/分，常称为阵发性心动过速。心率低于 60 次/分者（一般在 40 次/分以上），称为窦性心动过缓，可见于长期从事重体力劳动和运动员；病理性的见于甲状腺机能低下、颅内压增高、阻塞性黄疸以及洋地黄、奎尼丁或心得安类药物过量或中毒。如心率低于 40 次/分，应考虑有房室传导阻滞。心率过快超过 160 次/分，或心率过慢低于 40 次/分，大多见于心脏病病人，病人常有心悸、胸闷、心前区不适等症状，应及早进行详细检查，以便针对病因进行治疗。心脏每次收缩时由心室向动脉输出的血量叫做每搏输出量，心脏每分钟输出的血量叫做每分输出量，正常人在安静状态下每搏输出量为 70 毫升，如果心率按每分钟 75 次计算，每分输出量约为 5250 毫升。心输出量的多少，是衡量心脏工作能力的一项指标。

16.2　开发一个 Android 版心率计

知识点讲解：光盘:视频\视频讲解\第 16 章\开发一个 Android 版心率计.avi

本实例的功能是在 Android 系统中开发一个心率测试应用程序。本实例是通过低功耗蓝牙（BLE）技术

来检测传感器的变化的，根据检测到的变化数据测试心率的变化。本应用程序实例会搜索附近的 BLE 设备，并连接到远程心脏速率传感器的服务，会监听信号（平均脉冲）和心脏心率的变化（心跳或 RR 数据），能够检测到呼吸率如何影响心率变异，并根据检测结果生成三维图形演示。在本实例中，使用 BLE 技术传输了心脏速率的数据。因为具有开放的协议，所以可以较快地建立一个开发环境。因为 BLE 是从 Android 4.3 开始推出的，所以本项目需要运行在 Android 4.3 的设备上。

16.2.1　扫描蓝牙设备

本系统的主界面 Activity 是 DeviceScanActivity，对应界面布局文件是 actionbar_indeterminate_progress.xml，具体实现代码如下所示。

```
<FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_height="wrap_content"
        android:layout_width="56dp"
        android:minWidth="56dp">
    <ProgressBar
            android:layout_width="32dp"
            android:layout_height="32dp"
            android:layout_gravity="center"/>
</FrameLayout>
```

通过上述代码，在屏幕中显示进度控件来显示当前的扫描进度。

主界面 Activity 的实现文件是 DeviceScanActivity.java，功能是调用 UI 布局文件 actionbar_indeterminate_progress.xml 加载显示的控件，如果没有扫描到 BLE 蓝牙设备，则输出 error_bluetooth_not_supported 提示，如果搜索到，则列表显示蓝牙 BLE 设备。文件 DeviceScanActivity.java 的具体实现代码如下所示。

```
/**
 *扫描，并显示可用的蓝牙 BLE 设备
 */
public class DeviceScanActivity extends ListActivity {

    private static final int REQUEST_ENABLE_BT = 1;
    private static final long SCAN_PERIOD = 500;

    private BleDevicesAdapter leDeviceListAdapter;
    private BluetoothAdapter bluetoothAdapter;
    private Scanner scanner;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getActionBar().setTitle(R.string.title_devices);

        //使用此检查，以确定 BLE 是否支持在设备上，然后可以选择性地禁用 BLE 相关的功能
        if (!getPackageManager().hasSystemFeature(PackageManager.FEATURE_BLUETOOTH_LE)) {
            Toast.makeText(this, R.string.ble_not_supported, Toast.LENGTH_SHORT).show();
            finish();
            return;
        }
```

```
        //初始化一个蓝牙适配器。对于 API 级别 18 以上，通过 BluetoothManager 得到一个 BluetoothAdapter
        final BluetoothManager bluetoothManager =
                (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);
        bluetoothAdapter = bluetoothManager.getAdapter();

        // Checks if Bluetooth is supported on the device
        if (bluetoothAdapter == null) {
            Toast.makeText(this, R.string.error_bluetooth_not_supported, Toast.LENGTH_SHORT).show();
            finish();
            return;
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.gatt_scan, menu);
        if (scanner == null || !scanner.isScanning()) {
            menu.findItem(R.id.menu_stop).setVisible(false);
            menu.findItem(R.id.menu_scan).setVisible(true);
            menu.findItem(R.id.menu_refresh).setActionView(null);
        } else {
            menu.findItem(R.id.menu_stop).setVisible(true);
            menu.findItem(R.id.menu_scan).setVisible(false);
            menu.findItem(R.id.menu_refresh).setActionView(
                    R.layout.actionbar_indeterminate_progress);
        }
        return true;
    }
    //根据监听到的用户操作执行对应的事件处理程序
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menu_scan:
                leDeviceListAdapter.clear();
                if (scanner == null) {
                    scanner = new Scanner(bluetoothAdapter, mLeScanCallback);
                    scanner.startScanning();

                    invalidateOptionsMenu();
                }
                break;
            case R.id.menu_stop:
                if (scanner != null) {
                    scanner.stopScanning();
                    scanner = null;

                    invalidateOptionsMenu();
                }
                break;
        }
        return true;
```

```
}

@Override
protected void onResume() {
    super.onResume();

    // Ensures Bluetooth is enabled on the device.   If Bluetooth is not currently enabled,
    // fire an intent to display a dialog asking the user to grant permission to enable it.
    if (!bluetoothAdapter.isEnabled()) {
        final Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
        return;
    }

    init();
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // User chose not to enable Bluetooth
    if (requestCode == REQUEST_ENABLE_BT) {
        if (resultCode == Activity.RESULT_CANCELED) {
            finish();
        } else {
            init();
        }
    }
    super.onActivityResult(requestCode, resultCode, data);
}
//暂停扫描
@Override
protected void onPause() {
    super.onPause();

    if (scanner != null) {
        scanner.stopScanning();
        scanner = null;
    }
}
//监听用户单击操作命令选项事件
@Override
protected void onListItemClick(ListView l, View v, int position, long id) {
    final BluetoothDevice device = leDeviceListAdapter.getDevice(position);
    if (device == null)
        return;

    final Intent intent = new Intent(this, DeviceServicesActivity.class);
    intent.putExtra(DeviceServicesActivity.EXTRAS_DEVICE_NAME, device.getName());
    intent.putExtra(DeviceServicesActivity.EXTRAS_DEVICE_ADDRESS, device.getAddress());
    startActivity(intent);
}
```

```
private void init() {
    if (leDeviceListAdapter == null) {
        leDeviceListAdapter = new BleDevicesAdapter(getBaseContext());
        setListAdapter(leDeviceListAdapter);
    }

    if (scanner == null) {
        scanner = new Scanner(bluetoothAdapter, mLeScanCallback);
        scanner.startScanning();
    }

    invalidateOptionsMenu();
}

//扫描设备回调
private BluetoothAdapter.LeScanCallback mLeScanCallback =
        new BluetoothAdapter.LeScanCallback() {

            @Override
            public void onLeScan(final BluetoothDevice device, final int rssi, byte[ ] scanRecord) {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        leDeviceListAdapter.addDevice(device, rssi);
                        leDeviceListAdapter.notifyDataSetChanged();
                    }
                });
            }
        };

private static class Scanner extends Thread {
    private final BluetoothAdapter bluetoothAdapter;
    private final BluetoothAdapter.LeScanCallback mLeScanCallback;

    private volatile boolean isScanning = false;

    Scanner(BluetoothAdapter adapter, BluetoothAdapter.LeScanCallback callback) {
        bluetoothAdapter = adapter;
        mLeScanCallback = callback;
    }

    public boolean isScanning() {
        return isScanning;
    }

    public void startScanning() {
        synchronized (this) {
            isScanning = true;
            start();
        }
    }

    public void stopScanning() {
        synchronized (this) {
```

```
                isScanning = false;
                bluetoothAdapter.stopLeScan(mLeScanCallback);
            }
        }

        @Override
        public void run() {
            try {
                while (true) {
                    synchronized (this) {
                        if (!isScanning)
                            break;

                        bluetoothAdapter.startLeScan(mLeScanCallback);
                    }

                    sleep(SCAN_PERIOD);

                    synchronized (this) {
                        bluetoothAdapter.stopLeScan(mLeScanCallback);
                    }
                }
            } catch (InterruptedException ignore) {
            } finally {
                bluetoothAdapter.stopLeScan(mLeScanCallback);
            }
        }
    }
}
```

通过上述代码可知，在扫描过程中可以控制扫描操作，方法是单击设备右上角的 STOP、SCAN 按钮，执行效果如图 16-1 所示。



图 16-1　扫描到的蓝牙 BLE 设备列表

16.2.2 蓝牙控制界面

在主界面中列表显示扫描到的蓝牙 BLE 设备列表，单击列表中的某一个蓝牙设备后会来到蓝牙控制界面，在此界面可以设置和这个蓝牙设备的连接和断开连接操作，并显示蓝牙设备的 GAP 和 GATT 信息。蓝牙控制界面的 UI 布局文件是 gatt_services_characteristics.xml，具体实现代码如下所示。

```
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingTop="10dp">
    <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingLeft="10dp"
            android:paddingRight="10dp">
        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/label_device_address"
                android:textAppearance="?android:textAppearanceMedium"
                android:layout_marginRight="5dp"/>
        <TextView
                android:id="@+id/device_address"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textAppearance="?android:textAppearanceMedium"/>
    </LinearLayout>
    <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingLeft="10dp"
            android:paddingRight="10dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:textAppearanceMedium"
            android:text="@string/label_state"
            android:layout_marginRight="5dp"/>
        <TextView
            android:id="@+id/connection_state"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textAppearance="?android:textAppearanceMedium"/>
    </LinearLayout>
    <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
```

```
            android:layout_height="wrap_content"
            android:paddingLeft="10dp"
            android:paddingRight="10dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:textAppearanceMedium"
            android:text="@string/label_data"
            android:layout_marginRight="5dp"/>
        <TextView
            android:id="@+id/data_value"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textAppearance="?android:textAppearanceMedium"
            android:text="@string/no_data"
            android:minLines="3"/>
    </LinearLayout>
    <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingLeft="10dp"
            android:paddingRight="10dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:textAppearanceMedium"
            android:text="@string/label_heartrate"
            android:layout_marginRight="5dp"/>
        <TextView
            android:id="@+id/heartrate_value"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textAppearance="?android:textAppearanceMedium"
            android:text="@string/no_data"
            android:minLines="1"/>
    </LinearLayout>
    <LinearLayout
            android:orientation="horizontal"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingLeft="10dp"
            android:paddingRight="10dp">
        <Button
            android:id="@+id/demo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/action_demo"
            android:clickable="false"
            android:focusable="false" />
    </LinearLayout>
```

```
    <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/label_services"
            android:paddingTop="4dp"
            android:paddingBottom="4dp"
            android:paddingLeft="10dp"
            android:paddingRight="10dp"
            android:textStyle="bold"
            android:textAppearance="?android:textAppearanceMedium"
            android:background="@android:drawable/divider_horizontal_bright" />
    <ExpandableListView
            android:id="@+id/gatt_services_list"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:groupIndicator="@null"/>
</LinearLayout>
```

蓝牙控制界面的程序文件是 DeviceServicesActivity.java，功能是调用 UI 文件中的布局控件，并根据用户所选的蓝牙 BLE 设备提供了连接和显示数据界面，并显示了设备支持的 GATT 服务和特性。通过此 BleService 服务，可以实现与蓝牙 BLE 的 API 实现交互通信功能。

文件 DeviceServicesActivity.java 的具体实现代码如下所示。

```
public class DeviceServicesActivity extends Activity {
    private final static String TAG = DeviceServicesActivity.class.getSimpleName();

    public static final String EXTRAS_DEVICE_NAME = "DEVICE_NAME";
    public static final String EXTRAS_DEVICE_ADDRESS = "DEVICE_ADDRESS";

    private TextView connectionState;
    private TextView dataField;
    private TextView heartRateField;
    private TextView intervalField;
    private Button demoButton;

    private ExpandableListView gattServicesList;
    private BleServicesAdapter gattServiceAdapter;

    private String deviceName;
    private String deviceAddress;
    private BleService bleService;
    private boolean isConnected = false;

    private BleSensor<?> activeSensor;
    private BleSensor<?> heartRateSensor;

     private OnServiceItemClickListener serviceListener;

    //代码管理服务的生命周期
    private final ServiceConnection serviceConnection = new ServiceConnection() {

        @Override
```

```
        public void onServiceConnected(ComponentName componentName, IBinder service) {
            bleService = ((BleService.LocalBinder) service).getService();
            if (!bleService.initialize()) {
                Log.e(TAG, "Unable to initialize Bluetooth");
                finish();
            }
            // Automatically connects to the device upon successful start-up initialization.
            bleService.connect(deviceAddress);
        }

        @Override
        public void onServiceDisconnected(ComponentName componentName) {
            bleService = null;
        }
    };

    //处理蓝牙发射服务中的常见活动
    // ACTION_GATT_CONNECTED：连接一个 GATT 服务
    // ACTION_GATT_DISCONNECTED：断开一个 GATT 服务连接
    // ACTION_GATT_SERVICES_DISCOVERED：发现一个 GATT 服务
    // ACTION_DATA_AVAILABLE: received data from the device. This can be a result of read or notification
operations
    private final BroadcastReceiver gattUpdateReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            final String action = intent.getAction();
            if (BleService.ACTION_GATT_CONNECTED.equals(action)) {
                isConnected = true;
                updateConnectionState(R.string.connected);
                invalidateOptionsMenu();
            } else if (BleService.ACTION_GATT_DISCONNECTED.equals(action)) {
                isConnected = false;
                updateConnectionState(R.string.disconnected);
                invalidateOptionsMenu();
                clearUI();
            } else if (BleService.ACTION_GATT_SERVICES_DISCOVERED.equals(action)) {
                // Show all the supported services and characteristics on the user interface
                displayGattServices(bleService.getSupportedGattServices());
                  enableHeartRateSensor();
            } else if (BleService.ACTION_DATA_AVAILABLE.equals(action)) {
                  displayData(intent.getStringExtra(BleService.EXTRA_SERVICE_UUID),
intent.getStringExtra(BleService.EXTRA_TEXT));

            }
        }
    };

    // 如果给定了一个 GATT 服务选项，请检查所对应的支持功能
    // 具体请参考 http://d.android.com/reference/android/bluetooth/BluetoothGatt.html
    // 列出的支持的特性功能列表
```

```
    private final ExpandableListView.OnChildClickListener servicesListClickListner =
            new ExpandableListView.OnChildClickListener() {
                @Override
                public boolean onChildClick(ExpandableListView parent, View v, int groupPosition,
                                            int childPosition, long id) {
                    if (gattServiceAdapter == null)
                        return false;
                    final BluetoothGattCharacteristic characteristic = gattServiceAdapter.getChild(group
                    Position, childPosition);
                    final BleSensor<?> sensor = BleSensors.getSensor(characteristic.
                    getService().getUuid().toString());
                    if (activeSensor != null)
                        bleService.enableSensor(activeSensor, false);
                    if (sensor == null) {
                        bleService.readCharacteristic(characteristic);
                        return true;
                    }
                    if (sensor == activeSensor)
                        return true;
                    activeSensor = sensor;
                    bleService.enableSensor(sensor, true);
                    return true;
                }
            };
    private final BleServicesAdapter.OnServiceItemClickListener demoClickListener = new
BleServicesAdapter.OnServiceItemClickListener() {
        @Override
        public void onDemoClick(BluetoothGattService service) {
          Log.d(TAG, "onDemoClick: service" +service.getUuid().toString());
            final BleSensor<?> sensor = BleSensors.getSensor(service.getUuid().toString());
            if (sensor == null)
                return;
            final Class<? extends DemoSensorActivity> demoClass;
            if (sensor instanceof BleHeartRateSensor)
                demoClass = DemoHeartRateSensorActivity.class;
            else
                return;
            final Intent demoIntent = new Intent();
            demoIntent.setClass(DeviceServicesActivity.this, demoClass);
            demoIntent.putExtra(DemoSensorActivity.EXTRAS_DEVICE_ADDRESS, deviceAddress);
            demoIntent.putExtra(DemoSensorActivity.EXTRAS_SENSOR_UUID,
service.getUuid().toString());
            startActivity(demoIntent);
        }
        @Override
        public void onServiceEnabled(BluetoothGattService service, boolean enabled) {
            if (gattServiceAdapter == null)
                return;
            final BleSensor<?> sensor = BleSensors.getSensor(service.getUuid().toString());
            if (sensor == null)
                return;
```

```
            if (sensor == activeSensor)
                return;
            if (activeSensor != null)
                bleService.enableSensor(activeSensor, false);
            activeSensor = sensor;
            bleService.enableSensor(sensor, true);
        }
        @Override
        public void onServiceUpdated(BluetoothGattService service) {
            final BleSensor<?> sensor = BleSensors.getSensor(service.getUuid().toString());
            if (sensor == null)
                return;
            bleService.updateSensor(sensor);
        }
    };
    private void clearUI() {
        gattServicesList.setAdapter((SimpleExpandableListAdapter) null);
        dataField.setText(R.string.no_data);
          heartRateField.setText(R.string.no_data);
          intervalField.setText(R.string.no_data);
    }
     public void setServiceListener(OnServiceItemClickListener listener) {
          this.serviceListener = listener;
     }
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.gatt_services_characteristics);
        final Intent intent = getIntent();
        deviceName = intent.getStringExtra(EXTRAS_DEVICE_NAME);
        deviceAddress = intent.getStringExtra(EXTRAS_DEVICE_ADDRESS);
        //设置用户界面
        ((TextView) findViewById(R.id.device_address)).setText(deviceAddress);
        gattServicesList = (ExpandableListView) findViewById(R.id.gatt_services_list);
        gattServicesList.setOnChildClickListener(servicesListClickListner);
        connectionState = (TextView) findViewById(R.id.connection_state);
        dataField = (TextView) findViewById(R.id.data_value);
          heartRateField = (TextView) findViewById(R.id.heartrate_value);
          demoButton = (Button) findViewById(R.id.demo);
          demoButton.setOnClickListener(new View.OnClickListener() {
              @Override
              public void onClick(View v) {
                  Log.d(TAG, "onClick serviceListener: "+serviceListener);
                  if (serviceListener == null)
                      return;
                  final BluetoothGattService service = gattServiceAdapter.getHeartRateService();
                  serviceListener.onDemoClick(service);
                  Log.d(TAG, "set service listener");
              }
          });
          demoButton.setVisibility(View.VISIBLE);
```

```
        getActionBar().setTitle(deviceName);
        getActionBar().setDisplayHomeAsUpEnabled(true);
        final Intent gattServiceIntent = new Intent(this, BleService.class);
        bindService(gattServiceIntent, serviceConnection, BIND_AUTO_CREATE);
    }
    @Override
    protected void onResume() {
        super.onResume();
        registerReceiver(gattUpdateReceiver, makeGattUpdateIntentFilter());
        if (bleService != null) {
            final boolean result = bleService.connect(deviceAddress);
            Log.d(TAG, "Connect request result=" + result);
        }
    }
    @Override
    protected void onPause() {
        super.onPause();
        unregisterReceiver(gattUpdateReceiver);
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        unbindService(serviceConnection);
        bleService = null;
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.gatt_services, menu);
        if (isConnected) {
            menu.findItem(R.id.menu_connect).setVisible(false);
            menu.findItem(R.id.menu_disconnect).setVisible(true);
        } else {
            menu.findItem(R.id.menu_connect).setVisible(true);
            menu.findItem(R.id.menu_disconnect).setVisible(false);
        }
        return true;
    }
    //根据用户选择设置连接状态
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()) {
            case R.id.menu_connect:
                bleService.connect(deviceAddress);
                return true;
            case R.id.menu_disconnect:
                bleService.disconnect();
                return true;
            case android.R.id.home:
                onBackPressed();
                return true;
```

```
        }
        return super.onOptionsItemSelected(item);
    }
    //更新连接状态
    private void updateConnectionState(final int resourceId) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                connectionState.setText(resourceId);
            }
        });
    }
    //显示数据
    private void displayData(String uuid, String data) {
          if (data != null) {
               if (uuid.equals(BleHeartRateSensor.getServiceUUIDString())) {
                    heartRateField.setText(data);
               } else {
                    dataField.setText(data);
               }
          }
    }
     private boolean enableHeartRateSensor() {
          if (gattServiceAdapter == null)
               return false;
          final BluetoothGattCharacteristic characteristic = gattServiceAdapter
                    .getHeartRateCharacteristic();
          Log.d(TAG,"characteristic: " + characteristic);
          final BleSensor<?> sensor = BleSensors.getSensor(characteristic
                    .getService()
                    .getUuid()
                    .toString());
          if (heartRateSensor != null)
               bleService.enableSensor(heartRateSensor, false);
          if (sensor == null) {
               bleService.readCharacteristic(characteristic);
               return true;
          }
          if (sensor == heartRateSensor)
               return true;
          heartRateSensor = sensor;
          bleService.enableSensor(sensor, true);

        this.setServiceListener(demoClickListener);
          return true;
     }

    private void displayGattServices(List<BluetoothGattService> gattServices) {
        if (gattServices == null)
            return;
        gattServiceAdapter = new BleServicesAdapter(this, gattServices);
```

```
            gattServiceAdapter.setServiceListener(demoClickListener);
            gattServicesList.setAdapter(gattServiceAdapter);
        }
        private static IntentFilter makeGattUpdateIntentFilter() {
            final IntentFilter intentFilter = new IntentFilter();
            intentFilter.addAction(BleService.ACTION_GATT_CONNECTED);
            intentFilter.addAction(BleService.ACTION_GATT_DISCONNECTED);
            intentFilter.addAction(BleService.ACTION_GATT_SERVICES_DISCOVERED);
            intentFilter.addAction(BleService.ACTION_DATA_AVAILABLE);
            return intentFilter;
        }
}
```

在上述代码中，可以控制当前设备的连接状态，此功能是通过文件 BleService.java 实现的，具体实现代码如下所示。

```
/**
 *服务管理连接，并与托管给定的蓝牙 BLE 设备上的 GATT 服务器进行数据通信
 */
public class BleService extends Service {
    private final static String TAG = BleService.class.getSimpleName();

    private BluetoothManager bluetoothManager;
    private BluetoothAdapter adapter;
    private String deviceAddress;
    private BluetoothGatt gatt;
    private int connectionState = STATE_DISCONNECTED;

    private static final int STATE_DISCONNECTED = 0;
    private static final int STATE_CONNECTING = 1;
    private static final int STATE_CONNECTED = 2;

    private final static String INTENT_PREFIX = BleService.class.getPackage().getName();
    public final static String ACTION_GATT_CONNECTED =
INTENT_PREFIX+".ACTION_GATT_CONNECTED";
    public final static String ACTION_GATT_DISCONNECTED =
INTENT_PREFIX+".ACTION_GATT_DISCONNECTED";
    public final static String ACTION_GATT_SERVICES_DISCOVERED =
INTENT_PREFIX+".ACTION_GATT_SERVICES_DISCOVERED";
    public final static String ACTION_DATA_AVAILABLE = INTENT_PREFIX+".ACTION_DATA_AVAILABLE";
    public final static String EXTRA_SERVICE_UUID = INTENT_PREFIX+".EXTRA_SERVICE_UUID";
    public final static String EXTRA_CHARACTERISTIC_UUID =
INTENT_PREFIX+".EXTRA_CHARACTERISTIC_UUI";
    public final static String EXTRA_DATA = INTENT_PREFIX+".EXTRA_DATA";
    public final static String EXTRA_TEXT = INTENT_PREFIX+".EXTRA_TEXT";

    //实现回调方法，用于处理 GATT 事件的应用程序，例如，连接变化和发现服务
    private final BluetoothGattExecutor executor = new BluetoothGattExecutor() {

        @Override
        public void onConnectionStateChange(BluetoothGatt gatt, int status, int newState) {
            super.onConnectionStateChange(gatt, status, newState);
```

```
        String intentAction;
        if (newState == BluetoothProfile.STATE_CONNECTED) {
            intentAction = ACTION_GATT_CONNECTED;
            connectionState = STATE_CONNECTED;
            broadcastUpdate(intentAction);
            Log.i(TAG, "Connected to GATT server.");
            // Attempts to discover services after successful connection
            Log.i(TAG, "Attempting to start service discovery:" +
                    BleService.this.gatt.discoverServices());
        } else if (newState == BluetoothProfile.STATE_DISCONNECTED) {
            intentAction = ACTION_GATT_DISCONNECTED;
            connectionState = STATE_DISCONNECTED;
            Log.i(TAG, "Disconnected from GATT server.");
            broadcastUpdate(intentAction);
        }
    }

    @Override
    public void onServicesDiscovered(BluetoothGatt gatt, int status) {
        super.onServicesDiscovered(gatt, status);

        if (status == BluetoothGatt.GATT_SUCCESS) {
            broadcastUpdate(ACTION_GATT_SERVICES_DISCOVERED);
        } else {
            Log.w(TAG, "onServicesDiscovered received: " + status);
        }
    }

    @Override
    public void onCharacteristicRead(BluetoothGatt gatt,
                                     BluetoothGattCharacteristic characteristic,
                                     int status) {
        super.onCharacteristicRead(gatt, characteristic, status);

        if (status == BluetoothGatt.GATT_SUCCESS) {
            final BleSensor<?> sensor = BleSensors.getSensor(characteristic.
            getService().getUuid().toString());
            if (sensor != null) {
                if (sensor.onCharacteristicRead(characteristic)) {
                    return;
                }
            }

            broadcastUpdate(ACTION_DATA_AVAILABLE, characteristic);
        }
    }

    @Override
    public void onCharacteristicChanged(BluetoothGatt gatt,
                                        BluetoothGattCharacteristic characteristic) {
```

```
            super.onCharacteristicChanged(gatt, characteristic);

            broadcastUpdate(ACTION_DATA_AVAILABLE, characteristic);
        }
    };

    private void broadcastUpdate(final String action) {
        final Intent intent = new Intent(action);
        sendBroadcast(intent);
    }

    private void broadcastUpdate(final String action,
                                 final BluetoothGattCharacteristic characteristic) {
        final Intent intent = new Intent(action);
        intent.putExtra(EXTRA_SERVICE_UUID, characteristic.getService().getUuid().toString());
        intent.putExtra(EXTRA_CHARACTERISTIC_UUID, characteristic.getUuid().toString());

        final BleSensor<?> sensor = BleSensors.getSensor(characteristic.getService().getUuid().toString());
        if (sensor != null) {
            sensor.onCharacteristicChanged(characteristic);
            final String text = sensor.getDataString();
            intent.putExtra(EXTRA_TEXT, text);
            sendBroadcast(intent);
        } else {
            //对于所有其他的配置文件，写入十六进制格式的数据
            final byte[ ] data = characteristic.getValue();
            if (data != null && data.length > 0) {
                final StringBuilder stringBuilder = new StringBuilder(data.length);
                for (byte byteChar : data)
                    stringBuilder.append(String.format("%02X ", byteChar));
                intent.putExtra(EXTRA_TEXT, new String(data) + "\n" + stringBuilder.toString());
            }
        }
        sendBroadcast(intent);
    }

    public class LocalBinder extends Binder {
        public BleService getService() {
            return BleService.this;
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mBinder;
    }

    @Override
    public boolean onUnbind(Intent intent) {
    //使用给定的设备后，应该确保 BluetoothGatt.close()被调用
     //这样可以正确清理资源。在这个特定的例子中
```

```
        //当 UI 断开服务时调用 close()
            close();
            return super.onUnbind(intent);
        }

        /**
         *启用或禁用对一个给定特性的通知
         *
         * @param sensor
         * @param enabled If true, enable notification.   False otherwise
         */
        public void enableSensor(BleSensor<?> sensor, boolean enabled) {
            if (sensor == null)
                return;

            if (adapter == null || gatt == null) {
                Log.w(TAG, "BluetoothAdapter not initialized");
                return;
            }

            executor.enable(sensor, enabled);
            executor.execute(gatt);
        }

        private final IBinder mBinder = new LocalBinder();

        /**
         *初始化一个引用到本地蓝牙适配器
         *
         * @return Return true if the initialization is successful
         */
        public boolean initialize() {
            // For API level 18 and above, get a reference to BluetoothAdapter through BluetoothManager
            if (bluetoothManager == null) {
                bluetoothManager = (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);
                if (bluetoothManager == null) {
                    Log.e(TAG, "Unable to initialize BluetoothManager.");
                    return false;
                }
            }

            adapter = bluetoothManager.getAdapter();
            if (adapter == null) {
                Log.e(TAG, "Unable to obtain a BluetoothAdapter.");
                return false;
            }

            return true;
        }

        /**
```

```
     *连接到托管蓝牙 BLE 设备上的 GATT 服务器
     *
     * @param address The device address of the destination device.
     *
     * @return Return true if the connection is initiated successfully. The connection result
     *         is reported asynchronously through the
     *         {@code BluetoothGattCallback#onConnectionStateChange(android.bluetooth.BluetoothGatt, int, int)}
     *         callback
     */
    public boolean connect(final String address) {
        if (adapter == null || address == null) {
            Log.w(TAG, "BluetoothAdapter not initialized or unspecified address.");
            return false;
        }

        //前面连接的设备。尝试重新连接
        if (deviceAddress != null && address.equals(deviceAddress)
                && gatt != null) {
            Log.d(TAG, "Trying to use an existing BluetoothGatt for connection.");
            if (gatt.connect()) {
                connectionState = STATE_CONNECTING;
                return true;
            } else {
                return false;
            }
        }

        final BluetoothDevice device = adapter.getRemoteDevice(address);
        if (device == null) {
            Log.w(TAG, "Device not found.  Unable to connect.");
            return false;
        }
        //因为要直接连接到设备上，所以设置了自动连接，参数设置为 false
        gatt = device.connectGatt(this, false, executor);
        Log.d(TAG, "Trying to create a new connection.");
        deviceAddress = address;
        connectionState = STATE_CONNECTING;
        return true;
    }

    /**
     *断开现有连接或取消一个挂起的连接。断线结果通过异步
     * {@code BluetoothGattCallback#onConnectionStateChange(android.bluetooth.BluetoothGatt, int, int)}
     * 回调
     */
    public void disconnect() {
        if (adapter == null || gatt == null) {
            Log.w(TAG, "BluetoothAdapter not initialized");
            return;
        }
        gatt.disconnect();
```

```
    }

    /**
     *使用给定的 BLE 设备后，应用程序必须调用此方法，以确保资源被正确释放
     */
    public void close() {
        if (gatt == null) {
            return;
        }
        gatt.close();
        gatt = null;
    }
    /**
     *读取一个给定的{@code BluetoothGattCharacteristic}. 读取结果通过异步
     *{@code BluetoothGattCallback#onCharacteristicRead(android.bluetooth.BluetoothGatt,
android.bluetooth.BluetoothGattCharacteristic, int)} 回调
     *
     * @param characteristic The characteristic to read from
     */
    public void readCharacteristic(BluetoothGattCharacteristic characteristic) {
        if (adapter == null || gatt == null) {
            Log.w(TAG, "BluetoothAdapter not initialized");
            return;
        }
        gatt.readCharacteristic(characteristic);
    }
    public void updateSensor(BleSensor<?> sensor) {
        if (sensor == null)
            return;
        if (adapter == null || gatt == null) {
            Log.w(TAG, "BluetoothAdapter not initialized");
            return;
        }
        executor.update(sensor);
        executor.execute(gatt);
    }
    /**
     *检索连接的设备上支持关 GATT 服务列表
     * @return A {@code List} of supported services
     */
    public List<BluetoothGattService> getSupportedGattServices() {
        if (gatt == null) return null;
        return gatt.getServices();
    }
}
```

在上述代码中，通过文件 BluetoothGattExecutor.java 实现了蓝牙 GATT 服务的具体操作，具体实现代码如下所示。

```
public class BluetoothGattExecutor extends BluetoothGattCallback {
    public interface ServiceAction {
        public static final ServiceAction NULL = new ServiceAction() {
```

```
            @Override
            public boolean execute(BluetoothGatt bluetoothGatt) {
                // It is null action. do nothing
                return true;
            }
        };
        /***
         * 执行操作
         * @param bluetoothGatt
         * @return true - if action was executed instantly. false if action is waiting for
         *          feedback
         */
        public boolean execute(BluetoothGatt bluetoothGatt);
    }
    private final LinkedList<BluetoothGattExecutor.ServiceAction> queue = new LinkedList<ServiceAction>();
    private volatile ServiceAction currentAction;
    public void update(final BleSensor sensor) {
        queue.add(sensor.update());
    }
    public void enable(BleSensor sensor, boolean enable) {
        final ServiceAction[ ] actions = sensor.enable(enable);
        for ( ServiceAction action : actions ) {
            this.queue.add(action);
        }
    }
    public void execute(BluetoothGatt gatt) {
        if (currentAction != null)
            return;
        boolean next = !queue.isEmpty();
        while (next) {
            final BluetoothGattExecutor.ServiceAction action = queue.pop();
            currentAction = action;
            if (!action.execute(gatt))
                break;
            currentAction = null;
            next = !queue.isEmpty();
        }
    }
    @Override
    public void onDescriptorWrite(BluetoothGatt gatt, BluetoothGattDescriptor descriptor, int status) {
        super.onDescriptorWrite(gatt, descriptor, status);
        currentAction = null;
        execute(gatt);
    }
    @Override
    public void onCharacteristicWrite(BluetoothGatt gatt, BluetoothGattCharacteristic characteristic, int status) {
        super.onCharacteristicWrite(gatt, characteristic, status);
        currentAction = null;
        execute(gatt);
    }
    @Override
```

```
    public void onConnectionStateChange(BluetoothGatt gatt, int status, int newState) {
        if (newState == BluetoothProfile.STATE_DISCONNECTED) {
            queue.clear();
        }
    }
    @Override
    public void onCharacteristicRead(BluetoothGatt gatt,
                                      BluetoothGattCharacteristic characteristic,
                                      int status) {
        currentAction = null;
        execute(gatt);
    }
}
```

蓝牙设置界面执行效果如图 16-2 所示。

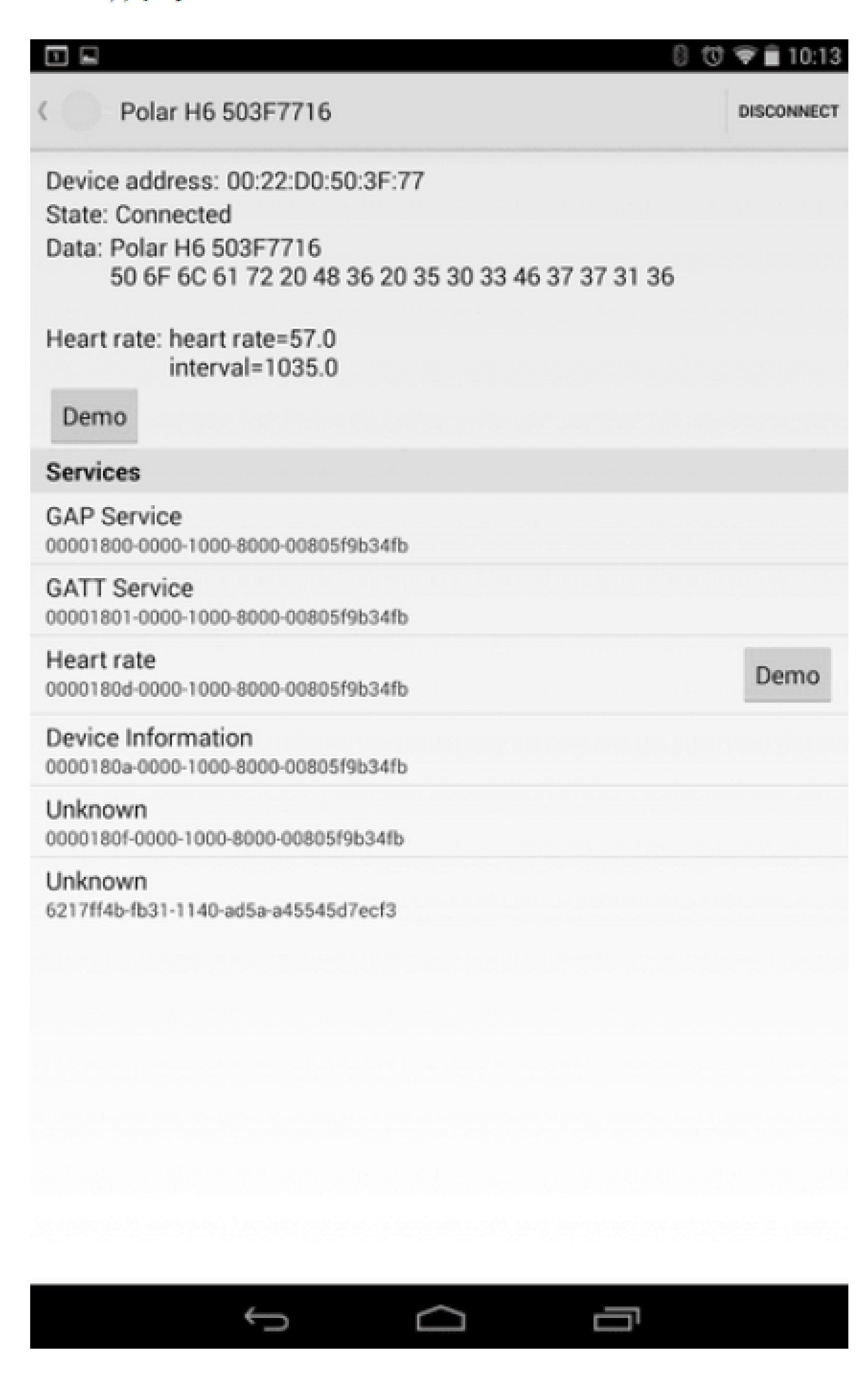


图 16-2　某个蓝牙设备的控制界面

16.2.3　蓝牙 BLE 设备适配器

16.2.2 节中用到了蓝牙 BLE 设备适配器功能，通过适配器列表显示了当前扫描到的蓝牙 BLE 设备。蓝牙 BLE 设备适配器的布局文件是 listitem_device.xml，具体实现代码如下所示。

```
<RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="4dp"
        android:paddingBottom="4dp"
```

```
            android:paddingRight="10dp"
            android:paddingLeft="10dp">
      <TextView
                  android:id="@+id/device_rssi"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_alignParentRight="true"
                  android:layout_centerVertical="true"/>

      <LinearLayout
                  android:orientation="vertical"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:layout_toLeftOf="@id/device_rssi">
            <TextView
                  android:id="@+id/device_name"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:textAppearance="?android:textAppearanceMedium"
                  android:textStyle="bold"/>
            <TextView
                  android:id="@+id/device_address"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"/>
      </LinearLayout>
</RelativeLayout>
```

对应的程序文件是 BleDevicesAdapter.java，功能是扫描附近的 BLE 蓝牙设备，具体实现代码如下所示。

```
public class BleDevicesAdapter extends BaseAdapter {
      private final LayoutInflater inflater;

      private final ArrayList<BluetoothDevice> leDevices;
      private final HashMap<BluetoothDevice, Integer> rssiMap = new HashMap<BluetoothDevice, Integer>();

      public BleDevicesAdapter(Context context) {
            leDevices = new ArrayList<BluetoothDevice>();
            inflater = LayoutInflater.from(context);
      }

      public void addDevice(BluetoothDevice device, int rssi) {
            if (!leDevices.contains(device)) {
                  leDevices.add(device);
            }
            rssiMap.put(device, rssi);
      }

      public BluetoothDevice getDevice(int position) {
            return leDevices.get(position);
      }

      public void clear() {
            leDevices.clear();
```

```
    }

    @Override
    public int getCount() {
        return leDevices.size();
    }

    @Override
    public Object getItem(int i) {
        return leDevices.get(i);
    }

    @Override
    public long getItemId(int i) {
        return i;
    }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        ViewHolder viewHolder;
        // General ListView optimization code
        if (view == null) {
            view = inflater.inflate(R.layout.listitem_device, null);
            viewHolder = new ViewHolder();
            viewHolder.deviceAddress = (TextView) view.findViewById(R.id.device_address);
            viewHolder.deviceName = (TextView) view.findViewById(R.id.device_name);
            viewHolder.deviceRssi = (TextView) view.findViewById(R.id.device_rssi);
            view.setTag(viewHolder);
        } else {
            viewHolder = (ViewHolder) view.getTag();
        }

        BluetoothDevice device = leDevices.get(i);
        final String deviceName = device.getName();
        if (deviceName != null && deviceName.length() > 0)
            viewHolder.deviceName.setText(deviceName);
        else
            viewHolder.deviceName.setText(R.string.unknown_device);
        viewHolder.deviceAddress.setText(device.getAddress());
        viewHolder.deviceRssi.setText(""+rssiMap.get(device)+" dBm");

        return view;
    }

    private static class ViewHolder {
        TextView deviceName;
        TextView deviceAddress;
        TextView deviceRssi;
    }
}
```

16.2.4 蓝牙 BLE 服务适配器

16.2.2 节中用到了蓝牙 BLE 服务适配器功能，通过适配器列表显示了当前扫描到的蓝牙 BLE 服务。蓝牙 BLE 服务适配器的布局文件是 listitem_service.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="4dp"
        android:paddingBottom="4dp"
        android:paddingRight="10dp"
        android:paddingLeft="10dp">

    <Button
            android:id="@+id/demo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_centerVertical="true"
            android:text="@string/action_demo"
            android:clickable="false"
            android:focusable="false" />
    <TextView
            android:id="@+id/name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_toLeftOf="@id/modes"
            android:textAppearance="?android:textAppearanceMedium" />
    <TextView
            android:id="@+id/uuid"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_toLeftOf="@id/modes"
            android:layout_below="@+id/name" />

</RelativeLayout>
```

对应的程序文件是 BleServicesAdapter.java，具体实现代码如下所示。

```
public class BleServicesAdapter extends BaseExpandableListAdapter {

     private final static String TAG = BleServicesAdapter.class.getSimpleName();

     public interface OnServiceItemClickListener {
          public void onDemoClick(BluetoothGattService service);

          public void onServiceEnabled(BluetoothGattService service,
                    boolean enabled);

          public void onServiceUpdated(BluetoothGattService service);
     }
```

```
private static final String MODE_READ = "R";
private static final String MODE_NOTIFY = "N";
private static final String MODE_WRITE = "W";

private final ArrayList<BluetoothGattService> services;
private final HashMap<BluetoothGattService, ArrayList<BluetoothGattCharacteristic>> characteristics;
private final LayoutInflater inflater;

private BluetoothGattService heartRateService;
private BluetoothGattCharacteristic heartRateCharacteristic;

private OnServiceItemClickListener serviceListener;

public BleServicesAdapter(Context context,
        List<BluetoothGattService> gattServices) {
        inflater = LayoutInflater.from(context);

        services = new ArrayList<BluetoothGattService>(gattServices.size());
        characteristics = new HashMap<BluetoothGattService, ArrayList<BluetoothGattCharacteristic>>(
                        gattServices.size());
        for (BluetoothGattService gattService : gattServices) {
                final List<BluetoothGattCharacteristic> gattCharacteristics = gattService
                                .getCharacteristics();
                characteristics.put(gattService,
                                new ArrayList<BluetoothGattCharacteristic>(
                                                gattCharacteristics));
                services.add(gattService);

                if (gattService.getUuid().equals(
                                UUID.fromString(BleHeartRateSensor.getServiceUUIDString()))) {
                        heartRateService = gattService;
                        heartRateCharacteristic = gattCharacteristics.get(0);
                }
        }
}

public void setServiceListener(OnServiceItemClickListener listener) {
        this.serviceListener = listener;
}

@Override
public int getGroupCount() {
        return services.size();
}

@Override
public int getChildrenCount(int groupPosition) {
        return characteristics.get(getGroup(groupPosition)).size();
}
```

```
@Override
public BluetoothGattService getGroup(int groupPosition) {
    return services.get(groupPosition);
}

@Override
public BluetoothGattCharacteristic getChild(int groupPosition,
        int childPosition) {
    Log.d(TAG, "group:" + groupPosition + " child:" + childPosition);
    Log.d(TAG, "uuid:" + characteristics.get(getGroup(groupPosition))
                            .get(childPosition).getUuid());
    return characteristics.get(getGroup(groupPosition)).get(childPosition);
}

public BluetoothGattService getHeartRateService() {
    return heartRateService;
}

public BluetoothGattCharacteristic getHeartRateCharacteristic() {
    return heartRateCharacteristic;
}

@Override
public long getGroupId(int groupPosition) {
    return groupPosition;
}

@Override
public long getChildId(int groupPosition, int childPosition) {
    return groupPosition * 100 + childPosition;
}

@Override
public boolean hasStableIds() {
    return true;
}

@Override
public View getGroupView(int groupPosition, boolean isExpanded,
        View convertView, ViewGroup parent) {
    final GroupViewHolder holder;
    if (convertView == null) {
        holder = new GroupViewHolder();

        convertView = inflater.inflate(R.layout.listitem_service, parent, false);
        holder.name = (TextView) convertView.findViewById(R.id.name);
        holder.uuid = (TextView) convertView.findViewById(R.id.uuid);
        holder.demo = convertView.findViewById(R.id.demo);

        holder.demo.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
                public void onClick(View v) {
                    if (serviceListener == null)
                        return;
                    final BluetoothGattService service = (BluetoothGattService) holder.demo
                            .getTag();
                    serviceListener.onDemoClick(service);
                }
            });

            convertView.setTag(holder);
        } else {
            holder = (GroupViewHolder) convertView.getTag();
        }

        final BluetoothGattService item = getGroup(groupPosition);

        final String uuid = item.getUuid().toString();
        final BleSensor<?> sensor = BleSensors.getSensor(uuid);
        final BleInfoService infoService = BleInfoServices.getService(uuid);

        final String serviceName;

        if (sensor != null)
            serviceName = sensor.getName();
        else if (infoService != null)
            serviceName = infoService.getName();
        else
            serviceName = "Unknown";

        holder.name.setText(serviceName);
        holder.uuid.setText(uuid);
        if (isDemoable(sensor)) {
            holder.demo.setTag(item);
            holder.demo.setVisibility(View.VISIBLE);
        } else {
            holder.demo.setVisibility(View.GONE);
        }

        return convertView;
    }

    @Override
    public View getChildView(int groupPosition, int childPosition,
            boolean isLastChild, View convertView, ViewGroup parent) {
        final ChildViewHolder holder;
        if (convertView == null) {
            holder = new ChildViewHolder();

            convertView = inflater.inflate(R.layout.listitem_characteristic,
                    parent, false);
            holder.name = (TextView) convertView.findViewById(R.id.name);
```

```
            holder.uuid = (TextView) convertView.findViewById(R.id.uuid);
            holder.modes = (TextView) convertView.findViewById(R.id.modes);
            holder.seek = (SeekBar) convertView.findViewById(R.id.seek);
            holder.seek
                    .setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
                        @Override
                        public void onProgressChanged(SeekBar seekBar,
                                int progress, boolean fromUser) {
                            if (serviceListener == null || !fromUser)
                                return;

                            final BleSensor<?> sensor = BleSensors
                                    .getSensor(holder.service.getUuid()
                                            .toString());
                            if (sensor == null)
                                return;

                        }

                        @Override
                        public void onStartTrackingTouch(SeekBar seekBar) {
                        }

                        @Override
                        public void onStopTrackingTouch(SeekBar seekBar) {
                        }
                    });

            convertView.setTag(holder);
        } else {
            holder = (ChildViewHolder) convertView.getTag();
        }

        final BluetoothGattCharacteristic item = getChild(groupPosition,
                childPosition);

        final String uuid = item.getUuid().toString();
        final String name;
        final String modes = getModeString(item.getProperties());

        holder.service = item.getService();

        final String serviceUUID = item.getService().getUuid().toString();
        final BleSensor<?> sensor = BleSensors.getSensor(serviceUUID);
        final BleInfoService infoService = BleInfoServices
                .getService(serviceUUID);

        if (sensor != null) {
            name = sensor.getCharacteristicName(uuid);

            if (sensor.isConfigUUID(uuid)) {
```

```
            } else {
                holder.uuid.setVisibility(View.VISIBLE);
                holder.seek.setVisibility(View.GONE);
            }
        } else if (infoService != null) {
            name = infoService.getCharacteristicName(uuid);

            holder.uuid.setVisibility(View.VISIBLE);
            holder.seek.setVisibility(View.GONE);
        } else {
            name = "Unknown";

            holder.uuid.setVisibility(View.VISIBLE);
            holder.seek.setVisibility(View.GONE);
        }

        holder.name.setText(name);
        holder.uuid.setText(uuid);
        holder.modes.setText(modes);

        return convertView;
    }

    @Override
    public boolean isChildSelectable(int groupPosition, int childPosition) {
        return true;
    }

    private static boolean isDemoable(BleSensor<?> sensor) {
        if (sensor instanceof BleHeartRateSensor)
            return true;
        return false;
    }

    private static String getModeString(int prop) {
        final StringBuilder modeBuilder = new StringBuilder();
        if ((prop & BluetoothGattCharacteristic.PROPERTY_READ) > 0) {
            modeBuilder.append(MODE_READ);
        }
        if ((prop & BluetoothGattCharacteristic.PROPERTY_NOTIFY) > 0) {
            if (modeBuilder.length() > 0)
                modeBuilder.append("/");
            modeBuilder.append(MODE_NOTIFY);
        }
        if ((prop & BluetoothGattCharacteristic.PROPERTY_WRITE) > 0) {
            if (modeBuilder.length() > 0)
                modeBuilder.append("/");
            modeBuilder.append(MODE_WRITE);
        }
        return modeBuilder.toString();
```

```
	}

	private static class GroupViewHolder {
		public TextView name;
		public TextView uuid;
		public View demo;
	}

	private static class ChildViewHolder {
		public BluetoothGattService service;

		public TextView name;
		public TextView uuid;
		public TextView modes;
		public SeekBar seek;
	}
}
```

16.2.5 传感器测试心率

本实例的核心功能是通过传感器测试心率，此功能的核心文件是 BleHeartRateSensor.java，功能是获取当前蓝牙设备的信息参数，例如服务 UUID、数据 UUID 等，并输出获得心率的信息，输出的信息有 UINT16 和 UINT8 两种格式。

```
public class BleHeartRateSensor extends BleSensor<float[ ]> {
	private final static String TAG = BleHeartRateSensor.class.getSimpleName();
	private static final String UUID_SENSOR_BODY_LOCATION = "00002a38-0000-1000-8000-00805f9b34fb";
	private static final int SENSOR_BODY_LOCATION_OTHER = 0;
	private static final int SENSOR_BODY_LOCATION_CHEST = 1;
	private static final int SENSOR_BODY_LOCATION_WRIST = 2;
	private static final int SENSOR_BODY_LOCATION_FINGER = 3;
	private static final int SENSOR_BODY_LOCATION_HAND = 4;
	private static final int SENSOR_BODY_LOCATION_EAR = 5;
	private static final int SENSOR_BODY_LOCATION_FOOT = 6;
	private int location = -1;
	BleHeartRateSensor() {
		super();
	}
	@Override
	public String getName() {
		return "Heart rate";
	}
	@Override
	public String getServiceUUID() {
		return "0000180d-0000-1000-8000-00805f9b34fb";
	}

	public static String getServiceUUIDString() {
		return "0000180d-0000-1000-8000-00805f9b34fb";
	}
```

```
    @Override
    public String getDataUUID() {
        return "00002a37-0000-1000-8000-00805f9b34fb";
    }

    public static String getDataUUIDString() {
        return "00002a37-0000-1000-8000-00805f9b34fb";
    }

    @Override
    public String getConfigUUID() {
        return "00002902-0000-1000-8000-00805f9b34fb";
    }

   @Override
   public String getCharacteristicName(String uuid) {
       if (UUID_SENSOR_BODY_LOCATION.equals(uuid))
           return getName() + " Sensor body location";
       return super.getCharacteristicName(uuid);
   }

   @Override
   public boolean onCharacteristicRead(BluetoothGattCharacteristic c) {
       super.onCharacteristicRead(c);
       Log.d(TAG, "onCharacteristicsReas");

       if ( !c.getUuid().toString().equals(UUID_SENSOR_BODY_LOCATION) )
           return false;
       location = c.getProperties();
       Log.d(TAG, "Sensor body location: " + location);
       return true;
   }
    @Override
    public String getDataString() {
        final float[ ] data = getData();
        return "heart rate=" + data[0] + "\ninterval=" + data[1];
    }

    @Override
    public float[ ] parse(BluetoothGattCharacteristic c) {

        double heartRate = extractHeartRate(c);
        double contact = extractContact(c);
        double energy = extractEnergyExpended(c);
        Integer[ ] interval = extractBeatToBeatInterval(c);

        float[ ] result = null;
        if (interval != null) {
            result = new float[interval.length + 1];
        } else {
            result = new float[2];
```

```
                result[1] = -1.0f;
            }
            result[0] = (float) heartRate;

            if (interval != null) {
                for (int i = 0; i < interval.length; i++) {
                    result[i+1] = interval[i].floatValue();
                }
            }

            return result;
        }

        private static double extractHeartRate(
                BluetoothGattCharacteristic characteristic) {

            int flag = characteristic.getProperties();
            Log.d(TAG, "Heart rate flag: " + flag);
            int format = -1;
            //Heart rate bit number format
            if ((flag & 0x01) != 0) {
                format = BluetoothGattCharacteristic.FORMAT_UINT16;
                Log.d(TAG, "Heart rate format UINT16.");
            } else {
                format = BluetoothGattCharacteristic.FORMAT_UINT8;
                Log.d(TAG, "Heart rate format UINT8.");
            }
            final int heartRate = characteristic.getIntValue(format, 1);
            Log.d(TAG, String.format("Received heart rate: %d", heartRate));
            return heartRate;
        }

        private static double extractContact(
                BluetoothGattCharacteristic characteristic) {

            int flag = characteristic.getProperties();
            int format = -1;
            //传感器连接状态
            if ((flag & 0x02) != 0) {
                Log.d(TAG, "Heart rate sensor contact info exists");
                if ((flag & 0x04) != 0) {
                    Log.d(TAG, "Heart rate sensor contact is ON");
                } else {
                    Log.d(TAG, "Heart rate sensor contact is OFF");
                }
            } else   {
                Log.d(TAG, "Heart rate sensor contact info doesn't exists");
            }
            //final int heartRate = characteristic.getIntValue(format, 1);
            //Log.d(TAG, String.format("Received heart rate: %d", heartRate));
            return 0.0d;
```

```
    }

    private static double extractEnergyExpended(
                BluetoothGattCharacteristic characteristic) {

        int flag = characteristic.getProperties();
        int format = -1;
        //计算活力状态
        if ((flag & 0x08) != 0) {
            Log.d(TAG, "Heart rate energy calculation exists.");
        } else {
            Log.d(TAG, "Heart rate energy calculation doesn't exists.");
        }
        //final int heartRate = characteristic.getIntValue(format, 1);
        //Log.d(TAG, String.format("Received heart rate: %d", heartRate));
        return 0.0d;
    }

    private static Integer[ ] extractBeatToBeatInterval(
                BluetoothGattCharacteristic characteristic) {

       int flag = characteristic.getIntValue(BluetoothGattCharacteristic.FORMAT_UINT8, 0);
       int format = -1;
       int energy = -1;
       int offset = 1; // This depends on hear rate value format and if there is energy data
       int rr_count = 0;

       if ((flag & 0x01) != 0) {
           format = BluetoothGattCharacteristic.FORMAT_UINT16;
           Log.d(TAG, "Heart rate format UINT16.");
           offset = 3;
       } else {
           format = BluetoothGattCharacteristic.FORMAT_UINT8;
           Log.d(TAG, "Heart rate format UINT8.");
           offset = 2;
       }
       if ((flag & 0x08) != 0) {
           //目前卡路里
           energy = characteristic.getIntValue(BluetoothGattCharacteristic.FORMAT_UINT16, offset);
           offset += 2;
           Log.d(TAG, "Received energy: { }"+ energy);
       }
       if ((flag & 0x16) != 0){
           //stuff 值
           Log.d(TAG, "RR stuff found at offset: "+ offset);
           Log.d(TAG, "RR length: "+ (characteristic.getValue()).length);
           rr_count = ((characteristic.getValue()).length - offset) / 2;
           Log.d(TAG, "RR length: "+ (characteristic.getValue()).length);
           Log.d(TAG, "rr_count: "+ rr_count);
             if (rr_count > 0) {
                  Integer[ ] mRr_values = new Integer[rr_count];
```

```
                    for (int i = 0; i < rr_count; i++) {
                        mRr_values[i] = characteristic.getIntValue(
                                BluetoothGattCharacteristic.FORMAT_UINT16, offset);
                        offset += 2;
                        Log.d(TAG, "Received RR: " + mRr_values[i]);
                    }
                    return mRr_values;
                }
            }
            Log.d(TAG, "No RR data on this update: ");
            return null;
        }
}
```

16.2.6 图形化显示心率值

当读取到传感器的心率值后，会通过 BLE 蓝牙将数据传输到检测设备中，在 Android 系统中通过图形化界面方式显示检测到的心率值。上述功能的 UI 布局文件是 demo_opengl.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

    <com.sample.hrv.demo.DemoGLSurfaceView
            android:id="@+id/gl"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
    <TextView
            android:id="@+id/text"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:textAppearance="?android:textAppearanceMedium"/>
</FrameLayout>
```

对应的程序文件是 DemoHeartRateSensorActivity.java，功能是根据获取的心率值构建一个图形化的心率图，具体实现代码如下所示。

```
public class DemoHeartRateSensorActivity extends DemoSensorActivity {
    private final static String TAG = DemoHeartRateSensorActivity.class
            .getSimpleName();

    private TextView viewText;
    private PolygonRenderer renderer;

    private GLSurfaceView view;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.demo_opengl);
        view = (GLSurfaceView) findViewById(R.id.gl);

        getActionBar().setTitle(R.string.title_demo_heartrate);

        viewText = (TextView) findViewById(R.id.text);

        renderer = new PolygonRenderer(this);
        view.setRenderer(renderer);
        //view.setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);
        // Render when hear rate data is updated
        view.setRenderMode(GLSurfaceView.RENDERMODE_WHEN_DIRTY);
    }

    @Override
    public void onDataRecieved(BleSensor<?> sensor, String text) {
        if (sensor instanceof BleHeartRateSensor) {
            final BleHeartRateSensor heartSensor = (BleHeartRateSensor) sensor;
            float[ ] values = heartSensor.getData();
            renderer.setInterval(values);
            view.requestRender();

            viewText.setText(text);
        }
    }

    public abstract class AbstractRenderer implements GLSurfaceView.Renderer {

        public int[ ] getConfigSpec() {
            int[ ] configSpec = { EGL10.EGL_DEPTH_SIZE, 0, EGL10.EGL_NONE };
            return configSpec;
        }

        public void onSurfaceCreated(GL10 gl, EGLConfig eglConfig) {
            gl.glDisable(GL10.GL_DITHER);
            gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT, GL10.GL_FASTEST);
            gl.glClearColor(.5f, .5f, .5f, 1);
            gl.glShadeModel(GL10.GL_SMOOTH);
            gl.glEnable(GL10.GL_DEPTH_TEST);
        }

        public void onSurfaceChanged(GL10 gl, int w, int h) {
            gl.glViewport(0, 0, w, h);
            float ratio = (float) w / h;
            gl.glMatrixMode(GL10.GL_PROJECTION);
            gl.glLoadIdentity();
            gl.glFrustumf(-ratio, ratio, -1, 1, 3, 7);
        }

        public void onDrawFrame(GL10 gl) {
            gl.glDisable(GL10.GL_DITHER);
```

```
            gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
            gl.glMatrixMode(GL10.GL_MODELVIEW);
            gl.glLoadIdentity();
            GLU.gluLookAt(gl, 0, 0, -5, 0f, 0f, 0f, 0f, 1.0f, 0.0f);
            gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
            draw(gl);
        }

        protected abstract void draw(GL10 gl);
    }

    public class PolygonRenderer extends AbstractRenderer {
        private final String TAG = PolygonRenderer.class
                .getSimpleName();

        // Number of points or vertices we want to use
        private final static int VERTS = 4;

        // A raw native buffer to hold the point coordinates
        private FloatBuffer mFVertexBuffer;

        // A raw native buffer to hold indices allowing a reuse of points
        private ShortBuffer mIndexBuffer;

        private int numOfIndecies = 0;

        private long prevtime = SystemClock.uptimeMillis();

        private int sides = 32;

        private float[ ] interval = { 0, 0, 0 };
        private float previousInterval = 0;

        public void setInterval(float[ ] interval) {
            if (this.interval[1] >= 0 && interval[1] > 0) {
                this.previousInterval = this.interval[1];
            }
            this.interval[0] = interval[0];      // heart rate
            this.interval[1] = interval[1];      // beat to beat interval
            this.interval[2] = 0;                // empty
        }

        public PolygonRenderer(Context context) {
            prepareBuffers(sides, interval[1]);
        }

        private void prepareBuffers(int sides, float radius) {
            Log.d(TAG,"radius: "+radius +" previous: "+previousInterval);
            // Is it a valid value
            if (radius < 0) {
                radius = previousInterval;
```

```
            }

            // Double check if the previous value was valid
            if (radius < 0) {
                radius = 700;
            }
            Log.d(TAG,"final radius: "+radius);

            radius = ( ( radius / 1000 ) - 0.7f ) * 2;

            RegularPolygon t = new RegularPolygon(0, 0, 0, radius, sides);
            this.mFVertexBuffer = t.getVertexBuffer();
            this.mIndexBuffer = t.getIndexBuffer();
            this.numOfIndecies = t.getNumberOfIndecies();
            this.mFVertexBuffer.position(0);
            this.mIndexBuffer.position(0);
        }
        // overriden method
        protected void draw(GL10 gl) {
            long curtime = SystemClock.uptimeMillis();
            this.prepareBuffers(sides, interval[1]);
            gl.glColor4f(96/255.0f, 246/255.0f, 255/255.0f, 1.0f);
            gl.glVertexPointer(3, GL10.GL_FLOAT, 0, mFVertexBuffer);
            gl.glDrawElements(GL10.GL_TRIANGLES, this.numOfIndecies,
                    GL10.GL_UNSIGNED_SHORT, mIndexBuffer);
        }
    }
    private static class RegularPolygon {
        private static final String TAG = RegularPolygon.class
                .getSimpleName();
        private float cx, cy, cz, r;
        private int sides;
        private float[ ] xarray = null;
        private float[ ] yarray = null;
        public RegularPolygon(float incx, float incy, float incz,     // (x,y,z)
                                                                      // center
                float inr, // radius
                int insides) // number of sides
        {
            cx = incx;
            cy = incy;
            cz = incz;
            r = inr;
            sides = insides;
            xarray = new float[sides];
            yarray = new float[sides];
            calcArrays();
        }
        private void calcArrays() {
            float[ ] xmarray = this.getXMultiplierArray();
            float[ ] ymarray = this.getYMultiplierArray();
```

```
        // calc xarray
        for (int i = 0; i < sides; i++) {
            float curm = xmarray[i];
            float xcoord = cx + r * curm;
            xarray[i] = xcoord;
        }
        //this.printArray(xarray, "xarray");
        // calc yarray
        for (int i = 0; i < sides; i++) {
            float curm = ymarray[i];
            float ycoord = cy + r * curm;
            yarray[i] = ycoord;
        }
        //this.printArray(yarray, "yarray");
    }
    public FloatBuffer getVertexBuffer() {
        int vertices = sides + 1;
        int coordinates = 3;
        int floatsize = 4;
        int spacePerVertex = coordinates * floatsize;

        ByteBuffer vbb = ByteBuffer.allocateDirect(spacePerVertex
                * vertices);
        vbb.order(ByteOrder.nativeOrder());
        FloatBuffer mFVertexBuffer = vbb.asFloatBuffer();

        // Put the first coordinate (x,y,z:0,0,0)
        mFVertexBuffer.put(0.0f); // x
        mFVertexBuffer.put(0.0f); // y
        mFVertexBuffer.put(0.0f); // z

        int totalPuts = 3;
        for (int i = 0; i < sides; i++) {
            mFVertexBuffer.put(xarray[i]); // x
            mFVertexBuffer.put(yarray[i]); // y
            mFVertexBuffer.put(0.0f); // z
            totalPuts += 3;
        }
        //Log.d(TAG, "total puts: " + Integer.toString(totalPuts));
        return mFVertexBuffer;
    }
    public ShortBuffer getIndexBuffer() {
        short[ ] iarray = new short[sides * 3];
        ByteBuffer ibb = ByteBuffer.allocateDirect(sides * 3 * 2);
        ibb.order(ByteOrder.nativeOrder());
        ShortBuffer mIndexBuffer = ibb.asShortBuffer();
        for (int i = 0; i < sides; i++) {
            short index1 = 0;
            short index2 = (short) (i + 1);
            short index3 = (short) (i + 2);
            if (index3 == sides + 1) {
```

```
                index3 = 1;
            }
            mIndexBuffer.put(index1);
            mIndexBuffer.put(index2);
            mIndexBuffer.put(index3);

            iarray[i * 3 + 0] = index1;
            iarray[i * 3 + 1] = index2;
            iarray[i * 3 + 2] = index3;
        }
        //this.printShortArray(iarray, "index array");
        return mIndexBuffer;
    }
    private float[ ] getXMultiplierArray() {
        float[ ] angleArray = getAngleArrays();
        float[ ] xmultiplierArray = new float[sides];
        for (int i = 0; i < angleArray.length; i++) {
            float curAngle = angleArray[i];
            float sinvalue = (float) Math.cos(Math.toRadians(curAngle));
            float absSinValue = Math.abs(sinvalue);
            if (isXPositiveQuadrant(curAngle)) {
                sinvalue = absSinValue;
            } else {
                sinvalue = -absSinValue;
            }
            xmultiplierArray[i] = this.getApproxValue(sinvalue);
        }
        //this.printArray(xmultiplierArray, "xmultiplierArray");
        return xmultiplierArray;
    }
    private float[ ] getYMultiplierArray() {
        float[ ] angleArray = getAngleArrays();
        float[ ] ymultiplierArray = new float[sides];
        for (int i = 0; i < angleArray.length; i++) {
            float curAngle = angleArray[i];
            float sinvalue = (float) Math.sin(Math.toRadians(curAngle));
            float absSinValue = Math.abs(sinvalue);
            if (isYPositiveQuadrant(curAngle)) {
                sinvalue = absSinValue;
            } else {
                sinvalue = -absSinValue;
            }
            ymultiplierArray[i] = this.getApproxValue(sinvalue);
        }
        //this.printArray(ymultiplierArray, "ymultiplierArray");
        return ymultiplierArray;
    }
    private boolean isXPositiveQuadrant(float angle) {
        if ((0 <= angle) && (angle <= 90)) {
            return true;
```

```
        }
        if ((angle < 0) && (angle >= -90)) {
            return true;
        }
        return false;
    }
    private boolean isYPositiveQuadrant(float angle) {
        if ((0 <= angle) && (angle <= 90)) {
            return true;
        }
        if ((angle < 180) && (angle >= 90)) {
            return true;
        }
        return false;
    }
    private float[ ] getAngleArrays() {
        float[ ] angleArray = new float[sides];
        float commonAngle = 360.0f / sides;
        float halfAngle = commonAngle / 2.0f;
        float firstAngle = 360.0f - (90 + halfAngle);
        angleArray[0] = firstAngle;
        float curAngle = firstAngle;
        for (int i = 1; i < sides; i++) {
            float newAngle = curAngle - commonAngle;
            angleArray[i] = newAngle;
            curAngle = newAngle;
        }
        //printArray(angleArray, "angleArray");
        return angleArray;
    }
    private float getApproxValue(float f) {
        if (Math.abs(f) < 0.001) {
            return 0;
        }
        return f;
    }
    public int getNumberOfIndecies() {
        return sides * 3;
    }
    private void printArray(float array[ ], String tag) {
        StringBuilder sb = new StringBuilder(tag);
        for (int i = 0; i < array.length; i++) {
            sb.append(";").append(array[i]);
        }
        Log.d(TAG, sb.toString());
    }
    private void printShortArray(short array[ ], String tag) {
        StringBuilder sb = new StringBuilder(tag);
        for (int i = 0; i < array.length; i++) {
            sb.append(";").append(array[i]);
```

```
                }
                Log.d(TAG, sb.toString());
            }
        }
}
```

执行效果如图 16-3 所示。

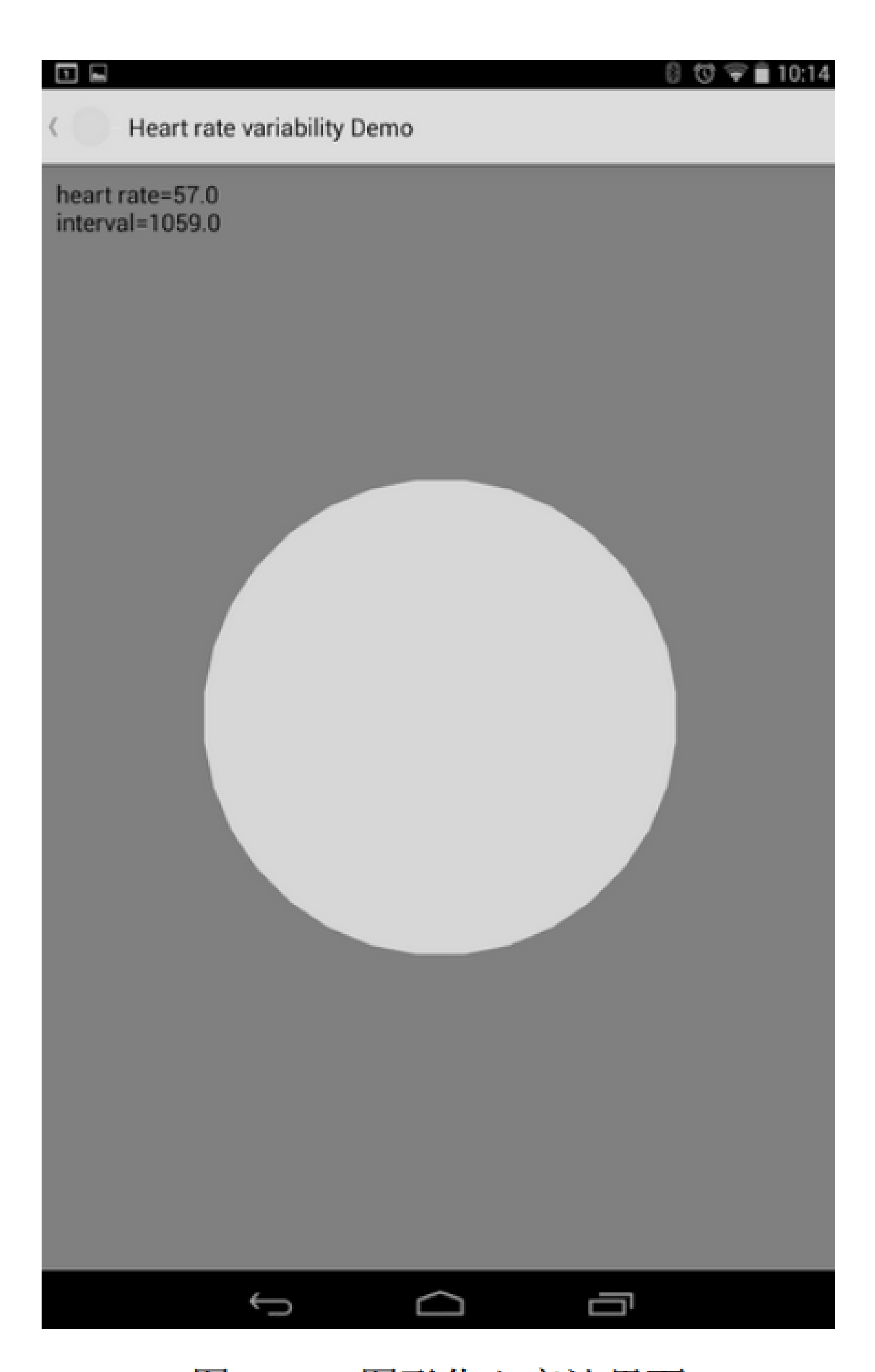


图 16-3　图形化心率计界面

第 17 章　仿陌陌交友系统

交友是人们为了摆脱自己单身的生活，而去结交他人的过程。交友的类型可以是女朋友或者男朋友，也可以是普通朋友。本章将详细讲解在 Android 系统中开发一款仿陌陌系统的交友软件，为读者掌握 Android 应用开发的核心技术打下基础。

17.1　陌 陌 介 绍

知识点讲解：光盘:视频\视频讲解\第 17 章\陌陌介绍.avi

陌陌是一款基于地理位置的移动社交工具，通过陌陌可以认识周围任意范围内的陌生人，查看对方的个人信息和位置，免费发送短信、语音、照片以及精准的地理位置。陌陌专注于移动互联网、移动社交和社交模式探索，并满足人们的社交愿望。公司于 2011 年 3 月份成立，由前网易总编辑唐岩创建，联合创始人及核心团队来自网易、新浪、凤凰网等公司。

17.1.1　陌陌发展现状

陌陌是陌陌科技开发的首个基于 iPhone 和 Android、Windows Phone 的手机应用，有别于微信、微博、QQ、YY、MSN、群群、遇见等手机社交软件。通过陌陌可以提供真实的位置信息，解决了以往社交软件过于虚幻，缺乏真实的线下互动的问题。

17.1.2　特点介绍

（1）社交模式

根据 GPS 搜寻和定位身边的陌生人和群组，高效快捷地建立联系，节省沟通的距离成本。

（2）免费传递

可以方便地通过陌陌免费发送短信、语音、照片以及精准的地理位置，与其他用户互动。

（3）体贴递送

即时了解信息送达的状态，“送达”、“已读”等提示能让用户及时掌握信息是否被对方看到。

（4）个人资料

可以在资料页存放 8 张照片，以及签名、职业、爱好等，以增进了解。

（5）场景表情

表情商店提供丰富的表情贴纸，让聊天过程更加生动，符合移动社交的聊天习惯。

（6）会员服务

可享受陌陌不断推出的各种增值及专属服务，包括基础会员服务、上限提升服务、表情商店服务等。

（7）隐私保护

可以随时把已添加的用户拉入黑名单，还可以对不良行为进行举报，并且有多种隐身模式。

（8）平台支持

全面支持 iPhone3G/3GS 及其以上版本的苹果设备，以及 Android 2.3 及以上版本的手机，支持各种网络接入方式。

17.2　实现系统欢迎界面

知识点讲解：光盘:视频\视频讲解\第 17 章\实现系统欢迎界面.avi

运行本系统后，将首先显示一个欢迎界面，以一幅图片作为背景，下方显示“注册”和“登录”按钮，如图 17-1 所示。



图 17-1　系统欢迎界面

本节将详细讲解系统欢迎界面的具体实现过程。

17.2.1　欢迎界面布局

本系统欢迎界面 Activity 是 WelcomeActivity，对应界面布局文件是 activity_welcome.xml，功能是通过 ImageView 控件显示背景图片，在界面下方通过两个 Button 控件显示“注册”和“登录”按钮，具体实现代码如下所示。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/pic_index_background"
    android:orientation="vertical" >

    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```

```
        android:layout_alignParentTop="true" >

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:scaleType="center"
            android:src="@drawable/pic_index_logo" />

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_alignParentTop="true"
            android:scaleType="center"
            android:visibility="gone" />
    </RelativeLayout>

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:scaleType="center"
        android:src="@drawable/pic_index_copyright" />

    <LinearLayout
        android:id="@+id/welcome_linear_ctrlbar"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:background="@drawable/bg_welcome_ctrlbar"
        android:gravity="center_horizontal|bottom"
        android:orientation="vertical"
        android:paddingBottom="15dip"
        android:paddingLeft="5dip"
        android:paddingRight="5dip"
        android:paddingTop="13dip" >

        <LinearLayout
            android:id="@+id/welcome_linear_avatars"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:orientation="horizontal" >

            <include
                android:id="@+id/welcome_include_member_avatar_block0"
                android:layout_weight="1"
                layout="@layout/include_welcome_item" />

            <include
```

```
            android:id="@+id/welcome_include_member_avatar_block1"
            android:layout_weight="1"
            layout="@layout/include_welcome_item" />

        <include
            android:id="@+id/welcome_include_member_avatar_block2"
            android:layout_weight="1"
            layout="@layout/include_welcome_item" />

        <include
            android:id="@+id/welcome_include_member_avatar_block3"
            android:layout_weight="1"
            layout="@layout/include_welcome_item" />

        <include
            android:id="@+id/welcome_include_member_avatar_block4"
            android:layout_weight="1"
            layout="@layout/include_welcome_item" />

        <include
            android:id="@+id/welcome_include_member_avatar_block5"
            android:layout_weight="1"
            layout="@layout/include_welcome_item" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:orientation="horizontal"
        android:visibility="invisible" >

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:src="@drawable/ic_index_totaluser" />

        <com.immomo.momo.android.view.HandyTextView
            android:id="@+id/welcome_htv_usercount"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:layout_marginLeft="5dip"
            android:layout_marginRight="5dip"
            android:text="0"
            android:textColor="#FFFFFFFF"
            android:textSize="18sp" />

        <com.immomo.momo.android.view.HandyTextView
            android:layout_width="wrap_content"
```

```
                android:layout_height="wrap_content"
                android:layout_gravity="bottom"
                android:text="位用户在你身边"
                android:textColor="#FFFFFFFF"
                android:textSize="13sp"
                android:textStyle="bold" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:orientation="horizontal" >

            <Button
                android:id="@+id/welcome_btn_register"
                android:layout_width="100dip"
                android:layout_height="40dip"
                android:layout_margin="5dip"
                android:background="@drawable/btn_default_blue"
                android:text="注册"
                android:textColor="#FFFFFFFF" />

            <Button
                android:id="@+id/welcome_btn_login"
                android:layout_width="100dip"
                android:layout_height="40dip"
                android:layout_margin="5dip"
                android:background="@drawable/btn_default_white"
                android:text="登录"
                android:textColor="#ff465579" />

            <ImageButton
                android:id="@+id/welcome_ibtn_about"
                android:layout_width="wrap_content"
                android:layout_height="40dip"
                android:layout_margin="5dip"
                android:layout_marginLeft="10dip"
                android:background="@drawable/btn_default_white"
                android:src="@drawable/ic_welcome_about_normal" />
        </LinearLayout>
    </LinearLayout>
```

17.2.2 欢迎界面 Activity

欢迎界面 Activity 的实现文件是 WelcomeActivity.java，功能是监听用户单击屏幕操作，根据用户单击的图标或按钮来到注册界面、登录界面或帮助界面。文件 WelcomeActivity.java 的具体实现代码如下所示。

```
public class WelcomeActivity extends BaseActivity implements OnClickListener {

    private LinearLayout mLinearCtrlbar;
```



```
private LinearLayout mLinearAvatars;
private Button mBtnRegister;
private Button mBtnLogin;
private ImageButton mIbtnAbout;

private View[ ] mMemberBlocks;
private String[ ] mAvatars = new String[ ] { "welcome_0", "welcome_1",
            "welcome_2", "welcome_3", "welcome_4", "welcome_5" };
private String[ ] mDistances = new String[ ] { "0.84km", "1.02km", "1.34km",
            "1.88km", "2.50km", "2.78km" };

@Override
protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_welcome);
    initViews();
    initEvents();
    initAvatarsItem();
    showWelcomeAnimation();
}

@Override
protected void initViews() {
    mLinearCtrlbar = (LinearLayout) findViewById(R.id.welcome_linear_ctrlbar);
    mLinearAvatars = (LinearLayout) findViewById(R.id.welcome_linear_avatars);
    mBtnRegister = (Button) findViewById(R.id.welcome_btn_register);
    mBtnLogin = (Button) findViewById(R.id.welcome_btn_login);
    mIbtnAbout = (ImageButton) findViewById(R.id.welcome_ibtn_about);
}

@Override
protected void initEvents() {
    mBtnRegister.setOnClickListener(this);
    mBtnLogin.setOnClickListener(this);
    mIbtnAbout.setOnClickListener(this);
}

private void initAvatarsItem() {
    initMemberBlocks();
    for (int i = 0; i < mMemberBlocks.length; i++) {
        ((ImageView) mMemberBlocks[i]
                .findViewById(R.id.welcome_item_iv_avatar))
                .setImageBitmap(mApplication.getAvatar(mAvatars[i]));
        ((HandyTextView) mMemberBlocks[i]
                .findViewById(R.id.welcome_item_htv_distance))
                .setText(mDistances[i]);
    }
}

private void initMemberBlocks() {
```

```
        mMemberBlocks = new View[6];
        mMemberBlocks[0] = findViewById(R.id.welcome_include_member_avatar_block0);
        mMemberBlocks[1] = findViewById(R.id.welcome_include_member_avatar_block1);
        mMemberBlocks[2] = findViewById(R.id.welcome_include_member_avatar_block2);
        mMemberBlocks[3] = findViewById(R.id.welcome_include_member_avatar_block3);
        mMemberBlocks[4] = findViewById(R.id.welcome_include_member_avatar_block4);
        mMemberBlocks[5] = findViewById(R.id.welcome_include_member_avatar_block5);

        int margin = (int) TypedValue.applyDimension(
                TypedValue.COMPLEX_UNIT_DIP, 4, getResources()
                        .getDisplayMetrics());
        int widthAndHeight = (mScreenWidth - margin * 12) / 6;
        for (int i = 0; i < mMemberBlocks.length; i++) {
            ViewGroup.LayoutParams params = mMemberBlocks[i].findViewById(
                    R.id.welcome_item_iv_avatar).getLayoutParams();
            params.width = widthAndHeight;
            params.height = widthAndHeight;
            mMemberBlocks[i].findViewById(R.id.welcome_item_iv_avatar)
                    .setLayoutParams(params);
        }
        mLinearAvatars.invalidate();
    }

    private void showWelcomeAnimation() {
        Animation animation = AnimationUtils.loadAnimation(
                WelcomeActivity.this, R.anim.welcome_ctrlbar_slideup);
        animation.setAnimationListener(new AnimationListener() {

            @Override
            public void onAnimationStart(Animation animation) {
                mLinearAvatars.setVisibility(View.GONE);
            }

            @Override
            public void onAnimationRepeat(Animation animation) {

            }

            @Override
            public void onAnimationEnd(Animation animation) {
                new Handler().postDelayed(new Runnable() {

                    @Override
                    public void run() {
                        mLinearAvatars.setVisibility(View.VISIBLE);
                    }
                }, 800);
            }
        });
        mLinearCtrlbar.startAnimation(animation);
    }
```

```
        @Override
        public void onClick(View v) {
                switch (v.getId()) {

                case R.id.welcome_btn_register:
                        startActivity(RegisterActivity.class);
                        break;

                case R.id.welcome_btn_login:
                        startActivity(LoginActivity.class);
                        break;

                case R.id.welcome_ibtn_about:
                        startActivity(AboutTabsActivity.class);
                        break;
                }
        }
}
```

17.3　实现系统注册界面

知识点讲解：光盘:视频\视频讲解\第 17 章\实现系统注册界面.avi

当在欢迎界面单击“注册”按钮后会来到系统注册界面，如图 17-2 所示。



图 17-2　系统注册界面

在本节的内容中，将详细讲解系统注册界面的具体实现过程。

17.3.1　注册界面布局

在系统注册界面中的布局文件是 activity_register.xml，功能是在上方显示注册表单供用户输入手机号码，

在下方显示“返回”和“下一步”按钮。文件 activity_register.xml 的具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/background_normal"
    android:orientation="vertical" >

    <include
        android:id="@+id/reg_header"
        layout="@layout/include_header" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_below="@+id/reg_header"
        android:orientation="vertical" >

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"
            android:orientation="vertical" >

            <ViewFlipper
                android:id="@+id/reg_vf_viewflipper"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:flipInterval="1000"
                android:persistentDrawingCache="animation" >

                <include
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    layout="@layout/include_register_phone" />

                <include
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    layout="@layout/include_register_verify" />

                <include
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    layout="@layout/include_register_setpwd" />

                <include
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    layout="@layout/include_register_baseinfo" />
```

```
                <include
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    layout="@layout/include_register_birthday" />

                <include
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    layout="@layout/include_register_photo" />
            </ViewFlipper>
        </LinearLayout>

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:background="@drawable/bg_unlogin_bar"
            android:gravity="center_vertical"
            android:orientation="horizontal"
            android:paddingBottom="4dip"
            android:paddingLeft="8dip"
            android:paddingRight="8dip"
            android:paddingTop="4dip" >

            <Button
                android:id="@+id/reg_btn_previous"
                android:layout_width="wrap_content"
                android:layout_height="42dip"
                android:layout_marginRight="9dip"
                android:layout_weight="1"
                android:background="@drawable/btn_bottombar"
                android:gravity="center"
                android:textColor="@color/profile_bottom_text_color"
                android:textSize="14sp" />

            <Button
                android:id="@+id/reg_btn_next"
                android:layout_width="wrap_content"
                android:layout_height="42dip"
                android:layout_marginLeft="9dip"
                android:layout_weight="1"
                android:background="@drawable/btn_bottombar"
                android:gravity="center"
                android:textColor="@color/profile_bottom_text_color"
                android:textSize="14sp" />
        </LinearLayout>
    </LinearLayout>

    <ImageView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/reg_header"
```

```
        android:background="@drawable/bg_topbar_shadow"
        android:focusable="true" />

</RelativeLayout>
```

17.3.2 注册界面 Activity

注册界面 Activity 的实现文件是 RegisterActivity.java，功能是监听用户单击屏幕操作，根据用户在表单中输入的注册信息进行验证。本系统设置的合法手机号码是 12345678901，如果输入其他号码，会输出“已经注册”的提示。文件 RegisterActivity.java 的具体实现代码如下所示。

```
public class RegisterActivity extends BaseActivity implements OnClickListener,
        onNextActionListener {

    private HeaderLayout mHeaderLayout;
    private ViewFlipper mVfFlipper;
    private Button mBtnPrevious;
    private Button mBtnNext;

    private BaseDialog mBackDialog;
    private RegisterStep mCurrentStep;
    private StepPhone mStepPhone;
    private StepVerify mStepVerify;
    private StepSetPassword mStepSetPassword;
    private StepBaseInfo mStepBaseInfo;
    private StepBirthday mStepBirthday;
    private StepPhoto mStepPhoto;

    private int mCurrentStepIndex = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        initViews();
        mCurrentStep = initStep();
        initEvents();
        initBackDialog();
    }

    @Override
    protected void onDestroy() {
        PhotoUtils.deleteImageFile();
        super.onDestroy();
    }

    @Override
    protected void initViews() {
        mHeaderLayout = (HeaderLayout) findViewById(R.id.reg_header);
        mHeaderLayout.init(HeaderStyle.TITLE_RIGHT_TEXT);
        mVfFlipper = (ViewFlipper) findViewById(R.id.reg_vf_viewflipper);
```

```
        mVfFlipper.setDisplayedChild(0);
        mBtnPrevious = (Button) findViewById(R.id.reg_btn_previous);
        mBtnNext = (Button) findViewById(R.id.reg_btn_next);
    }

    @Override
    protected void initEvents() {
        mCurrentStep.setOnNextActionListener(this);
        mBtnPrevious.setOnClickListener(this);
        mBtnNext.setOnClickListener(this);
    }

    @Override
    public void onBackPressed() {
        if (mCurrentStepIndex <= 1) {
            mBackDialog.show();
        } else {
            doPrevious();
        }
    }

    @Override
    public void onClick(View arg0) {
        switch (arg0.getId()) {
        case R.id.reg_btn_previous:
            if (mCurrentStepIndex <= 1) {
                mBackDialog.show();
            } else {
                doPrevious();
            }
            break;

        case R.id.reg_btn_next:
            if (mCurrentStepIndex < 6) {
                doNext();
            } else {
                if (mCurrentStep.validate()) {
                    mCurrentStep.doNext();
                }
            }
            break;
        }
    }

    @SuppressWarnings("deprecation")
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        switch (requestCode) {
        case PhotoUtils.INTENT_REQUEST_CODE_ALBUM:
            if (data == null) {
```

```
                return;
            }
            if (resultCode == RESULT_OK) {
                if (data.getData() == null) {
                    return;
                }
                if (!FileUtils.isSdcardExist()) {
                    showCustomToast("SD 卡不可用,请检查");
                    return;
                }
                Uri uri = data.getData();
                String[ ] proj = { MediaStore.Images.Media.DATA };
                Cursor cursor = managedQuery(uri, proj, null, null, null);
                if (cursor != null) {
                    int column_index = cursor
                            .getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
                    if (cursor.getCount() > 0 && cursor.moveToFirst()) {
                        String path = cursor.getString(column_index);
                        Bitmap bitmap = BitmapFactory.decodeFile(path);
                        if (PhotoUtils.bitmapIsLarge(bitmap)) {
                            PhotoUtils.cropPhoto(this, this, path);
                        } else {
                            mStepPhoto.setUserPhoto(bitmap);
                        }
                    }
                }
            }
            break;

        case PhotoUtils.INTENT_REQUEST_CODE_CAMERA:
            if (resultCode == RESULT_OK) {
                String path = mStepPhoto.getTakePicturePath();
                Bitmap bitmap = BitmapFactory.decodeFile(path);
                if (PhotoUtils.bitmapIsLarge(bitmap)) {
                    PhotoUtils.cropPhoto(this, this, path);
                } else {
                    mStepPhoto.setUserPhoto(bitmap);
                }
            }
            break;

        case PhotoUtils.INTENT_REQUEST_CODE_CROP:
            if (resultCode == RESULT_OK) {
                String path = data.getStringExtra("path");
                if (path != null) {
                    Bitmap bitmap = BitmapFactory.decodeFile(path);
                    if (bitmap != null) {
                        mStepPhoto.setUserPhoto(bitmap);
                    }
                }
            }
```

```
                break;
            }
        }

        @Override
        public void next() {
            mCurrentStepIndex++;
            mCurrentStep = initStep();
            mCurrentStep.setOnNextActionListener(this);
            mVfFlipper.setInAnimation(this, R.anim.push_left_in);
            mVfFlipper.setOutAnimation(this, R.anim.push_left_out);
            mVfFlipper.showNext();
        }

        private RegisterStep initStep() {
            switch (mCurrentStepIndex) {
            case 1:
                if (mStepPhone == null) {
                    mStepPhone = new StepPhone(this, mVfFlipper.getChildAt(0));
                }
                mHeaderLayout.setTitleRightText("注册新账号", null, "1/6");
                mBtnPrevious.setText("返回");
                mBtnNext.setText("下一步");
                return mStepPhone;

            case 2:
                if (mStepVerify == null) {
                    mStepVerify = new StepVerify(this, mVfFlipper.getChildAt(1));
                }
                mHeaderLayout.setTitleRightText("填写验证码", null, "2/6");
                mBtnPrevious.setText("上一步");
                mBtnNext.setText("下一步");
                return mStepVerify;

            case 3:
                if (mStepSetPassword == null) {
                    mStepSetPassword = new StepSetPassword(this,
                            mVfFlipper.getChildAt(2));
                }
                mHeaderLayout.setTitleRightText("设置密码", null, "3/6");
                mBtnPrevious.setText("上一步");
                mBtnNext.setText("下一步");
                return mStepSetPassword;

            case 4:
                if (mStepBaseInfo == null) {
                    mStepBaseInfo = new StepBaseInfo(this, mVfFlipper.getChildAt(3));
                }
                mHeaderLayout.setTitleRightText("填写基本资料", null, "4/6");
                mBtnPrevious.setText("上一步");
                mBtnNext.setText("下一步");
```

```
                return mStepBaseInfo;

            case 5:
                if (mStepBirthday == null) {
                    mStepBirthday = new StepBirthday(this, mVfFlipper.getChildAt(4));
                }
                mHeaderLayout.setTitleRightText("您的生日", null, "5/6");
                mBtnPrevious.setText("上一步");
                mBtnNext.setText("下一步");
                return mStepBirthday;

            case 6:
                if (mStepPhoto == null) {
                    mStepPhoto = new StepPhoto(this, mVfFlipper.getChildAt(5));
                }
                mHeaderLayout.setTitleRightText("设置头像", null, "6/6");
                mBtnPrevious.setText("上一步");
                mBtnNext.setText("注册");
                return mStepPhoto;
        }
        return null;
    }

    private void doPrevious() {
        mCurrentStepIndex--;
        mCurrentStep = initStep();
        mCurrentStep.setOnNextActionListener(this);
        mVfFlipper.setInAnimation(this, R.anim.push_right_in);
        mVfFlipper.setOutAnimation(this, R.anim.push_right_out);
        mVfFlipper.showPrevious();
    }

    private void doNext() {
        if (mCurrentStep.validate()) {
            if (mCurrentStep.isChange()) {
                mCurrentStep.doNext();
            } else {
                next();
            }
        }
    }

    private void initBackDialog() {
        mBackDialog = BaseDialog.getDialog(RegisterActivity.this, "提示",
                    "确认要放弃注册么?", "确认", new DialogInterface.OnClickListener() {

                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            dialog.dismiss();
                            finish();
                        }
```

```
                }, "取消", new DialogInterface.OnClickListener() {

                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                    }
                });
        mBackDialog.setButton1Background(R.drawable.btn_default_popsubmit);

    }

    @Override
    protected void putAsyncTask(AsyncTask<Void, Void, Boolean> asyncTask) {
        super.putAsyncTask(asyncTask);
    }

    @Override
    protected void showCustomToast(String text) {
        super.showCustomToast(text);
    }

    @Override
    protected void showLoadingDialog(String text) {
        super.showLoadingDialog(text);
    }

    @Override
    protected void dismissLoadingDialog() {
        super.dismissLoadingDialog();
    }

    protected int getScreenWidth() {
        return mScreenWidth;
    }

    protected BaseApplication getBaseApplication() {
        return mApplication;
    }

    protected String getPhoneNumber() {
        if (mStepPhone != null) {
            return mStepPhone.getPhoneNumber();
        }
        return "";
    }

}
```

如果注册手机号合法，则弹出输入验证码界面，如图 17-3 所示。



图 17-3　输入验证码界面

17.3.3　输入验证码界面 Activity

输入验证码界面 Activity 的实现文件是 StepVerify.java，功能是验证注册用户输入的验证码是否合法。在本项目中，设置的固定验证码是 123456。文件 StepVerify.java 的具体实现代码如下所示。

```
public class StepVerify extends RegisterStep implements OnClickListener,
        TextWatcher {

    private HandyTextView mHtvPhoneNumber;
    private EditText mEtVerifyCode;
    private Button mBtnResend;
    private HandyTextView mHtvNoCode;

    private static final String PROMPT = "验证码已经发送到* ";
    private static final String DEFAULT_VALIDATE_CODE = "123456";

    private boolean mIsChange = true;
    private String mVerifyCode;

    private int mReSendTime = 60;
    private BaseDialog mBaseDialog;

    public StepVerify(RegisterActivity activity, View contentRootView) {
        super(activity, contentRootView);
        handler.sendEmptyMessage(0);
    }

    @Override
    public void initViews() {
        mHtvPhoneNumber = (HandyTextView) findViewById(R.id.reg_verify_htv_phonenumber);
        mHtvPhoneNumber.setText(PROMPT + getPhoneNumber());
        mEtVerifyCode = (EditText) findViewById(R.id.reg_verify_et_verifycode);
        mBtnResend = (Button) findViewById(R.id.reg_verify_btn_resend);
```

```
            mBtnResend.setEnabled(false);
            mBtnResend.setText("重发(60)");
            mHtvNoCode = (HandyTextView) findViewById(R.id.reg_verify_htv_nocode);
            TextUtils.addUnderlineText(mContext, mHtvNoCode, 0, mHtvNoCode
                        .getText().toString().length());
//          TextUtils.addHyperlinks(mHtvNoCode, 0, mHtvNoCode
//                      .getText().toString().length(), new OnClickListener() {
//
//                          @Override
//                          public void onClick(View v) {
//                                System.out.println("123");
//                          }
//                      });
      }

      @Override
      public void initEvents() {
            mBtnResend.setOnClickListener(this);
            mHtvNoCode.setOnClickListener(this);
            mEtVerifyCode.addTextChangedListener(this);
      }

      @Override
      public void doNext() {
            putAsyncTask(new AsyncTask<Void, Void, Boolean>() {

                  @Override
                  protected void onPreExecute() {
                        super.onPreExecute();
                        showLoadingDialog("正在验证,请稍候...");
                  }

                  @Override
                  protected Boolean doInBackground(Void... params) {
                        try {
                              Thread.sleep(2000);
                              if (DEFAULT_VALIDATE_CODE.equals(mVerifyCode)) {
                                    return true;
                              }
                        } catch (InterruptedException e) {

                        }
                        return false;
                  }

                  @Override
                  protected void onPostExecute(Boolean result) {
                        super.onPostExecute(result);
                        dismissLoadingDialog();
                        if (result) {
                              mIsChange = false;
                              mOnNextActionListener.next();
```

```
                    } else {
                        mBaseDialog = BaseDialog.getDialog(mContext, "提示", "验证码错误",
                                "确认", new DialogInterface.OnClickListener() {

                                    @Override
                                    public void onClick(DialogInterface dialog,
                                            int which) {
                                        mEtVerifyCode.requestFocus();
                                        dialog.dismiss();
                                    }

                                });
                        mBaseDialog.show();
                    }
                }

            });
        }

        @Override
        public boolean validate() {
            if (isNull(mEtVerifyCode)) {
                showCustomToast("请输入验证码");
                mEtVerifyCode.requestFocus();
                return false;
            }
            mVerifyCode = mEtVerifyCode.getText().toString().trim();
            return true;
        }

        @Override
        public boolean isChange() {
            return mIsChange;
        }

        @Override
        public void onClick(View v) {
            switch (v.getId()) {
            case R.id.reg_verify_btn_resend:
                handler.sendEmptyMessage(0);
                break;

            case R.id.reg_verify_htv_nocode:
                showCustomToast("抱歉,暂时不支持此操作");
                break;
            }
        }

        @Override
        public void afterTextChanged(Editable s) {

        }
```

```
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count,
                int after) {

        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            mIsChange = true;
        }

        Handler handler = new Handler() {

            @Override
            public void handleMessage(Message msg) {
                super.handleMessage(msg);
                if (mReSendTime > 1) {
                    mReSendTime--;
                    mBtnResend.setEnabled(false);
                    mBtnResend.setText("重发(" + mReSendTime + ")");
                    handler.sendEmptyMessageDelayed(0, 1000);
                } else {
                    mReSendTime = 60;
                    mBtnResend.setEnabled(true);
                    mBtnResend.setText("重发");
                }
            }
        };
}
```

17.3.4　设置密码界面 Activity

如果输入的验证码合法，单击“下一步”按钮后会来到设置密码界面，在界面上方显示两个文本框供用户分别输入登录密码和确认密码，在界面下方显示“上一步”和“下一步”按钮，如图 17-4 所示。



图 17-4　设置密码界面

设置密码界面 Activity 的实现文件是 StepSetPassword.java，功能是验证注册用户输入的两个密码完全一致，并且是 6 位以上。文件 StepSetPassword.java 的具体实现代码如下所示。

```
public class StepSetPassword extends RegisterStep implements TextWatcher {
    private EditText mEtPwd;
    private EditText mEtRePwd;
    private boolean mIsChange = true;
    public StepSetPassword(RegisterActivity activity, View contentRootView) {
        super(activity, contentRootView);
    }
    @Override
    public void initViews() {
        mEtPwd = (EditText) findViewById(R.id.reg_setpwd_et_pwd);
        mEtRePwd = (EditText) findViewById(R.id.reg_setpwd_et_repwd);
    }

    @Override
    public void initEvents() {
        mEtPwd.addTextChangedListener(this);
        mEtRePwd.addTextChangedListener(this);
    }
    @Override
    public void doNext() {
        mIsChange = false;
        mOnNextActionListener.next();
    }
    @Override
    public boolean validate() {
        String pwd = null;
        String rePwd = null;
        if (isNull(mEtPwd)) {
            showCustomToast("请输入密码");
            mEtPwd.requestFocus();
            return false;
        } else {
            pwd = mEtPwd.getText().toString().trim();
            if (pwd.length() < 6) {
                showCustomToast("密码不能少于 6 位");
                mEtPwd.requestFocus();
                return false;
            }
        }
        if (isNull(mEtRePwd)) {
            showCustomToast("请重复输入一次密码");
            mEtRePwd.requestFocus();
            return false;
        } else {
            rePwd = mEtRePwd.getText().toString().trim();
            if (!pwd.equals(rePwd)) {
                showCustomToast("两次输入的密码不一致");
                mEtRePwd.requestFocus();
```

```
                    return false;
                }
            }
            return true;
        }
        @Override
        public boolean isChange() {
            return mIsChange;
        }
        @Override
        public void afterTextChanged(Editable s) {
        }
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count,
                int after) {
        }
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            mIsChange = true;
        }
}
```

17.3.5　设置用户名界面 Activity

如果输入的密码合法，单击“下一步”按钮后会来到设置用户名界面，在界面上方显示一个文本框供用户输入用户名，显示一个单选按钮供用户选择性别，在界面下方显示“上一步”和“下一步”按钮，如图 17-5 所示。



图 17-5　设置用户名界面

设置用户名界面 Activity 的实现文件是 StepBaseInfo.java，功能是验证是否输入用户名并选择性别。文

件 StepBaseInfo.java 的具体实现代码如下所示。

```
public class StepBaseInfo extends RegisterStep implements TextWatcher,
		OnCheckedChangeListener {

	private EditText mEtName;
	private RadioGroup mRgGender;
	private RadioButton mRbMale;
	private RadioButton mRbFemale;

	private boolean mIsChange = true;
	private boolean mIsGenderAlert;
	private BaseDialog mBaseDialog;

	public StepBaseInfo(RegisterActivity activity, View contentRootView) {
		super(activity, contentRootView);
	}

	@Override
	public void initViews() {
		mEtName = (EditText) findViewById(R.id.reg_baseinfo_et_name);
		mRgGender = (RadioGroup) findViewById(R.id.reg_baseinfo_rg_gender);
		mRbMale = (RadioButton) findViewById(R.id.reg_baseinfo_rb_male);
		mRbFemale = (RadioButton) findViewById(R.id.reg_baseinfo_rb_female);
	}

	@Override
	public void initEvents() {
		mEtName.addTextChangedListener(this);
		mRgGender.setOnCheckedChangeListener(this);
	}

	@Override
	public void doNext() {
		mOnNextActionListener.next();
	}

	@Override
	public boolean validate() {
		if (isNull(mEtName)) {
			showCustomToast("请输入用户名");
			mEtName.requestFocus();
			return false;
		}
		if (mRgGender.getCheckedRadioButtonId() < 0) {
			showCustomToast("请选择性别");
			return false;
		}
		return true;
	}

	@Override
```

```
	public boolean isChange() {
		return mIsChange;
	}

	@Override
	public void onCheckedChanged(RadioGroup group, int checkedId) {
		mIsChange = true;
		if (!mIsGenderAlert) {
			mIsGenderAlert = true;
			mBaseDialog = BaseDialog.getDialog(mContext, "提示", "注册成功后性别将不可更改",
					"确认", new DialogInterface.OnClickListener() {

						@Override
						public void onClick(DialogInterface dialog, int which) {
							dialog.dismiss();
						}
					});
			mBaseDialog.show();
		}
		switch (checkedId) {
		case R.id.reg_baseinfo_rb_male:
			mRbMale.setChecked(true);
			break;

		case R.id.reg_baseinfo_rb_female:
			mRbFemale.setChecked(true);
			break;
		}
	}
	@Override
	public void afterTextChanged(Editable s) {
	}
	@Override
	public void beforeTextChanged(CharSequence s, int start, int count,
			int after) {
	}
	@Override
	public void onTextChanged(CharSequence s, int start, int before, int count) {
		mIsChange = true;
	}
}
```

17.3.6　设置生日界面 Activity

如果设置的用户名和性别合法，单击“下一步”按钮后会来到设置生日界面，在界面上方显示年、月、日供用户选择生日，在界面下方显示“上一步”和“下一步”按钮，如图 17-6 所示。

设置生日界面 Activity 的实现文件是 StepBirthday.java，功能是验证用户设置的年龄的合法性，系统要求的合法年龄范围在 12～100 岁之间。文件 StepBirthday.java 的具体实现代码如下所示。



图 17-6　设置生日界面

```
public class StepBirthday extends RegisterStep implements OnDateChangedListener {

	private HandyTextView mHtvConstellation;
	private HandyTextView mHtvAge;
	private DatePicker mDpBirthday;

	private Calendar mCalendar;
	private Date mMinDate;
	private Date mMaxDate;
	private Date mSelectDate;
	private static final int MAX_AGE = 100;
	private static final int MIN_AGE = 12;

	public StepBirthday(RegisterActivity activity, View contentRootView) {
		super(activity, contentRootView);
		initData();

	}

	private void flushBirthday(Calendar calendar) {
		String constellation = TextUtils.getConstellation(
				calendar.get(Calendar.MONTH),
				calendar.get(Calendar.DAY_OF_MONTH));
		mSelectDate = calendar.getTime();
		mHtvConstellation.setText(constellation);
		int age = TextUtils.getAge(calendar.get(Calendar.YEAR),
				calendar.get(Calendar.MONTH),
				calendar.get(Calendar.DAY_OF_MONTH));
		mHtvAge.setText(age + "");
	}

	private void initData() {
```

```
        mSelectDate = DateUtils.getDate("19900101");

        Calendar mMinCalendar = Calendar.getInstance();
        Calendar mMaxCalendar = Calendar.getInstance();

        mMinCalendar.set(Calendar.YEAR, mMinCalendar.get(Calendar.YEAR)
                - MIN_AGE);
        mMinDate = mMinCalendar.getTime();
        mMaxCalendar.set(Calendar.YEAR, mMaxCalendar.get(Calendar.YEAR)
                - MAX_AGE);
        mMaxDate = mMaxCalendar.getTime();

        mCalendar = Calendar.getInstance();
        mCalendar.setTime(mSelectDate);
        flushBirthday(mCalendar);
        mDpBirthday.init(mCalendar.get(Calendar.YEAR),
                mCalendar.get(Calendar.MONTH),
                mCalendar.get(Calendar.DAY_OF_MONTH), this);
    }

    @Override
    public void initViews() {
        mHtvConstellation = (HandyTextView) findViewById(R.id.reg_birthday_htv_constellation);
        mHtvAge = (HandyTextView) findViewById(R.id.reg_birthday_htv_age);
        mDpBirthday = (DatePicker) findViewById(R.id.reg_birthday_dp_birthday);
    }

    @Override
    public void initEvents() {

    }

    @Override
    public void doNext() {
        mOnNextActionListener.next();
    }

    @Override
    public boolean validate() {
        return true;
    }

    @Override
    public boolean isChange() {
        return false;
    }

    @Override
    public void onDateChanged(DatePicker view, int year, int monthOfYear,
            int dayOfMonth) {
        mCalendar = Calendar.getInstance();
```

```
            mCalendar.set(year, monthOfYear, dayOfMonth);
            if (mCalendar.getTime().after(mMinDate)
                        || mCalendar.getTime().before(mMaxDate)) {
                mCalendar.setTime(mSelectDate);
                mDpBirthday.init(mCalendar.get(Calendar.YEAR),
                        mCalendar.get(Calendar.MONTH),
                        mCalendar.get(Calendar.DAY_OF_MONTH), this);
            } else {
                flushBirthday(mCalendar);
            }
        }

}
```

17.3.7 设置头像界面 Activity

如果设置的年龄合法，单击“下一步”按钮后会来到设置头像界面，在界面上方显示选择图片按钮和拍照按钮供用户快速设置头像，在界面下方显示“上一步”和“注册”按钮，如图 17-7 所示。



图 17-7 设置头像界面

设置头像界面 Activity 的实现文件是 StepPhoto.java，功能是验证用户是否设置了头像。文件 StepPhoto.java 的具体实现代码如下所示。

```
public class StepPhoto extends RegisterStep implements OnClickListener {

    private HandyTextView mHtvRecommendation;
    private ImageView mIvUserPhoto;
    private LinearLayout mLayoutSelectPhoto;
    private LinearLayout mLayoutTakePicture;
    private LinearLayout mLayoutAvatars;

    private View[ ] mMemberBlocks;
    private String[ ] mAvatars = new String[ ] { "welcome_0", "welcome_1",
                "welcome_2", "welcome_3", "welcome_4", "welcome_5" };
    private String[ ] mDistances = new String[ ] { "0.84km", "1.02km", "1.34km",
```

```
                "1.88km", "2.50km", "2.78km" };
    private String mTakePicturePath;
    private Bitmap mUserPhoto;

    private EditTextDialog mEditTextDialog;

    public StepPhoto(RegisterActivity activity, View contentRootView) {
        super(activity, contentRootView);
        initAvatarsItem();
    }

    private void initAvatarsItem() {
        initMemberBlocks();
        for (int i = 0; i < mMemberBlocks.length; i++) {
            ((ImageView) mMemberBlocks[i]
                    .findViewById(R.id.welcome_item_iv_avatar))
                    .setImageBitmap(getBaseApplication().getAvatar(mAvatars[i]));
            ((HandyTextView) mMemberBlocks[i]
                    .findViewById(R.id.welcome_item_htv_distance))
                    .setText(mDistances[i]);
        }
    }

    private void initMemberBlocks() {
        mMemberBlocks = new View[6];
        mMemberBlocks[0] = findViewById(R.id.reg_photo_include_member_avatar_block0);
        mMemberBlocks[1] = findViewById(R.id.reg_photo_include_member_avatar_block1);
        mMemberBlocks[2] = findViewById(R.id.reg_photo_include_member_avatar_block2);
        mMemberBlocks[3] = findViewById(R.id.reg_photo_include_member_avatar_block3);
        mMemberBlocks[4] = findViewById(R.id.reg_photo_include_member_avatar_block4);
        mMemberBlocks[5] = findViewById(R.id.reg_photo_include_member_avatar_block5);

        int margin = (int) TypedValue.applyDimension(
                TypedValue.COMPLEX_UNIT_DIP, 4, mContext.getResources()
                        .getDisplayMetrics());
        int widthAndHeight = (getScreenWidth() - margin * 12) / 6;
        for (int i = 0; i < mMemberBlocks.length; i++) {
            ViewGroup.LayoutParams params = mMemberBlocks[i].findViewById(
                    R.id.welcome_item_iv_avatar).getLayoutParams();
            params.width = widthAndHeight;
            params.height = widthAndHeight;
            mMemberBlocks[i].findViewById(R.id.welcome_item_iv_avatar)
                    .setLayoutParams(params);
        }
        mLayoutAvatars.invalidate();
    }

    public void setUserPhoto(Bitmap bitmap) {
        if (bitmap != null) {
            mUserPhoto = bitmap;
            mIvUserPhoto.setImageBitmap(mUserPhoto);
```

```
                return;
            }
            showCustomToast("未获取到图片");
            mUserPhoto = null;
            mIvUserPhoto.setImageResource(R.drawable.ic_common_def_header);
        }

        public String getTakePicturePath() {
            return mTakePicturePath;
        }

        @Override
        public void initViews() {
            mHtvRecommendation = (HandyTextView) findViewById(R.id.reg_photo_htv_recommendation);
            mIvUserPhoto = (ImageView) findViewById(R.id.reg_photo_iv_userphoto);
            mLayoutSelectPhoto = (LinearLayout) findViewById(R.id.reg_photo_layout_selectphoto);
            mLayoutTakePicture = (LinearLayout) findViewById(R.id.reg_photo_layout_takepicture);
            mLayoutAvatars = (LinearLayout) findViewById(R.id.reg_photo_layout_avatars);
        }

        @Override
        public void initEvents() {
            mHtvRecommendation.setOnClickListener(this);
            mLayoutSelectPhoto.setOnClickListener(this);
            mLayoutTakePicture.setOnClickListener(this);
        }

        @Override
        public boolean validate() {
            if (mUserPhoto == null) {
                showCustomToast("请添加头像");
                return false;
            }
            return true;
        }

        @Override
        public void doNext() {
            putAsyncTask(new AsyncTask<Void, Void, Boolean>() {

                @Override
                protected void onPreExecute() {
                    super.onPreExecute();
                    showLoadingDialog("请稍候，正在提交...");
                }

                @Override
                protected Boolean doInBackground(Void... params) {
                    try {
                        Thread.sleep(2000);
                        return true;
```

```
                } catch (InterruptedException e) {

                }
                return false;
            }

            @Override
            protected void onPostExecute(Boolean result) {
                super.onPostExecute(result);
                dismissLoadingDialog();
                if (result) {
                    mActivity.finish();
                }
            }

        });
    }

    @Override
    public boolean isChange() {
        return false;
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
        case R.id.reg_photo_htv_recommendation:
            mEditTextDialog = new EditTextDialog(mContext);
            mEditTextDialog.setTitle("填写推荐人");
            mEditTextDialog.setButton("取消",
                    new DialogInterface.OnClickListener() {

                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            mEditTextDialog.cancel();
                        }
                    }, "确认", new DialogInterface.OnClickListener() {

                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            String text = mEditTextDialog.getText();
                            if (text == null) {
                                mEditTextDialog.requestFocus();
                                showCustomToast("请输入推荐人号码");
                            } else {
                                mEditTextDialog.dismiss();
                                showCustomToast("您输入的推荐人号码为:" + text);
                            }
                        }
                    });
            mEditTextDialog.show();
```

```
                break;

            case R.id.reg_photo_layout_selectphoto:
                PhotoUtils.selectPhoto(mActivity);
                break;

            case R.id.reg_photo_layout_takepicture:
                mTakePicturePath = PhotoUtils.takePicture(mActivity);
                break;
            }
        }

}
```

设置头像完毕后，单击“注册”按钮完成注册。

17.4　实现系统主界面

知识点讲解：光盘:视频\视频讲解\第 17 章\实现系统主界面.avi

当用户输入合法的注册信息登录系统后，会首先显示系统主界面，如图 17-8 所示。



图 17-8　系统主界面

本节将详细讲解系统主界面的具体实现过程。

17.4.1　主界面布局

系统主界面的布局文件是 activity_maintabs.xml，功能是使用 TabWidget 控件将界面分割成 5 部分。文件 activity_maintabs.xml 的具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/tabhost"
```



```
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#ffffffff" >

        <RelativeLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" >

            <FrameLayout
                android:id="@android:id/tabcontent"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_above="@android:id/tabs"
                android:background="@color/background_normal" />

            <TabWidget
                android:id="@android:id/tabs"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_alignParentBottom="true"
                android:divider="@null" />
        </RelativeLayout>
    </LinearLayout>

</TabHost>
```

17.4.2 实现主界面 Activity

主界面 Activity 的实现文件是 MainTabActivity.java，功能是通过函数 initTabs()初始化显示 TabWidget 控件中的内容，默认设置显示“附近的人”。文件 MainTabActivity.java 的具体实现代码如下所示。

```
public class MainTabActivity extends TabActivity {
    private TabHost mTabHost;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maintabs);
        initViews();
        initTabs();
    }

    private void initViews() {
        mTabHost = getTabHost();
    }

    private void initTabs() {
```

```
            LayoutInflater inflater = LayoutInflater.from(MainTabActivity.this);

            View nearbyView = inflater.inflate(
                    R.layout.common_bottombar_tab_nearby, null);
            TabHost.TabSpec nearbyTabSpec = mTabHost.newTabSpec(
                    NearByActivity.class.getName()).setIndicator(nearbyView);
            nearbyTabSpec.setContent(new Intent(MainTabActivity.this,
                    NearByActivity.class));
            mTabHost.addTab(nearbyTabSpec);

            View nearbyFeedsView = inflater.inflate(
                    R.layout.common_bottombar_tab_site, null);
            TabHost.TabSpec nearbyFeedsTabSpec = mTabHost.newTabSpec(
                    NearByFeedsActivity.class.getName()).setIndicator(
                    nearbyFeedsView);
            nearbyFeedsTabSpec.setContent(new Intent(MainTabActivity.this,
                    NearByFeedsActivity.class));
            mTabHost.addTab(nearbyFeedsTabSpec);

            View sessionListView = inflater.inflate(
                    R.layout.common_bottombar_tab_chat, null);
            TabHost.TabSpec sessionListTabSpec = mTabHost.newTabSpec(
                    SessionListActivity.class.getName()).setIndicator(
                    sessionListView);
            sessionListTabSpec.setContent(new Intent(MainTabActivity.this,
                    SessionListActivity.class));
            mTabHost.addTab(sessionListTabSpec);

            View contactView = inflater.inflate(
                    R.layout.common_bottombar_tab_friend, null);
            TabHost.TabSpec contactTabSpec = mTabHost.newTabSpec(
                    ContactTabsActivity.class.getName()).setIndicator(contactView);
            contactTabSpec.setContent(new Intent(MainTabActivity.this,
                    ContactTabsActivity.class));
            mTabHost.addTab(contactTabSpec);

            View userSettingView = inflater.inflate(
                    R.layout.common_bottombar_tab_profile, null);
            TabHost.TabSpec userSettingTabSpec = mTabHost.newTabSpec(
                    UserSettingActivity.class.getName()).setIndicator(
                    userSettingView);
            userSettingTabSpec.setContent(new Intent(MainTabActivity.this,
                    UserSettingActivity.class));
            mTabHost.addTab(userSettingTabSpec);

        }
}
```

17.4.3 实现“附近的人”界面

在系统主界面中，中间大部分内容显示的是系统“附近的人”信息，此功能的实现布局文件是



common_bottombar_tab_nearby.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="0dip"
    android:layout_height="40dip"
    android:layout_weight="1"
    android:background="@drawable/bg_tb_item_center"
    android:paddingBottom="2dip" >

    <com.immomo.momo.android.view.HandyTextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:drawableTop="@drawable/ic_tab_nearby"
        android:gravity="center_horizontal"
        android:text="附近"
        android:textColor="@color/maintab_text_color"
        android:textSize="11sp"
        android:shadowDx="0.0"
        android:shadowDy="-1.0"
        android:shadowRadius="1.0"/>

</RelativeLayout>
```

“附近的人”界面 Activity 的实现文件是 NearByActivity.java，功能是在顶部显示“附近”、“群组”和“个人”选项卡，并监听用户单击屏幕事件，根据用户操作执行对应的事件处理函数。例如，单击搜索图标可以根据关键字快速检索附近的人。文件 NearByActivity.java 的具体实现代码如下所示。

```
public class NearByActivity extends TabItemActivity {

    private HeaderLayout mHeaderLayout;
    private HeaderSpinner mHeaderSpinner;
    private NearByPeopleFragment mPeopleFragment;
    private NearByGroupFragment mGroupFragment;

    private NearByPopupWindow mPopupWindow;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_nearby);
        initPopupWindow();
        initViews();
        initEvents();
        init();
    }

    @Override
    protected void initViews() {
        mHeaderLayout = (HeaderLayout) findViewById(R.id.nearby_header);
        mHeaderLayout.initSearch(new OnSearchClickListener());
        mHeaderSpinner = mHeaderLayout.setTitleNearBy("附近",
```

```
                        new OnSpinnerClickListener(), "附近群组",
                        R.drawable.ic_topbar_search,
                        new OnMiddleImageButtonClickListener(), "个人", "群组",
                        new OnSwitcherButtonClickListener());
        mHeaderLayout.init(HeaderStyle.TITLE_NEARBY_PEOPLE);
    }

    @Override
    protected void initEvents() {

    }

    @Override
    protected void init() {
        mPeopleFragment = new NearByPeopleFragment(mApplication, this, this);
        mGroupFragment = new NearByGroupFragment(mApplication, this, this);
        FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
        ft.replace(R.id.nearby_layout_content, mPeopleFragment).commit();
    }

    private void initPopupWindow() {
        mPopupWindow = new NearByPopupWindow(this);
        mPopupWindow.setOnSubmitClickListener(new onSubmitClickListener() {

            @Override
            public void onClick() {
                mPeopleFragment.onManualRefresh();
            }
        });
        mPopupWindow.setOnDismissListener(new OnDismissListener() {

            @Override
            public void onDismiss() {
                mHeaderSpinner.initSpinnerState(false);
            }
        });
    }

    public class OnSpinnerClickListener implements onSpinnerClickListener {

        @Override
        public void onClick(boolean isSelect) {
            if (isSelect) {
                mPopupWindow
                        .showViewTopCenter(findViewById(R.id.nearby_layout_root));
            } else {
                mPopupWindow.dismiss();
            }
        }
    }
```

```
public class OnSearchClickListener implements onSearchListener {

    @Override
    public void onSearch(EditText et) {
        String s = et.getText().toString().trim();
        if (TextUtils.isEmpty(s)) {
            showCustomToast("请输入搜索关键字");
            et.requestFocus();
        } else {
            ((InputMethodManager) getSystemService(INPUT_METHOD_SERVICE))
                    .hideSoftInputFromWindow(NearByActivity.this
                            .getCurrentFocus().getWindowToken(),
                            InputMethodManager.HIDE_NOT_ALWAYS);
            putAsyncTask(new AsyncTask<Void, Void, Boolean>() {

                @Override
                protected void onPreExecute() {
                    super.onPreExecute();
                    mHeaderLayout.changeSearchState(SearchState.SEARCH);
                }

                @Override
                protected Boolean doInBackground(Void... params) {
                    try {
                        Thread.sleep(2000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    return false;
                }

                @Override
                protected void onPostExecute(Boolean result) {
                    super.onPostExecute(result);
                    mHeaderLayout.changeSearchState(SearchState.INPUT);
                    showCustomToast("未找到搜索的群");
                }
            });
        }
    }

}

public class OnMiddleImageButtonClickListener implements
        onMiddleImageButtonClickListener {

    @Override
    public void onClick() {
        mHeaderLayout.showSearch();
    }
}
```

```
    public class OnSwitcherButtonClickListener implements
            onSwitcherButtonClickListener {

        @Override
        public void onClick(SwitcherButtonState state) {
            FragmentTransaction ft = getSupportFragmentManager()
                    .beginTransaction();
            ft.setCustomAnimations(R.anim.fragment_fadein,
                    R.anim.fragment_fadeout);
            switch (state) {
            case LEFT:
                mHeaderLayout.init(HeaderStyle.TITLE_NEARBY_PEOPLE);
                ft.replace(R.id.nearby_layout_content, mPeopleFragment)
                        .commit();
                break;

            case RIGHT:
                mHeaderLayout.init(HeaderStyle.TITLE_NEARBY_GROUP);
                ft.replace(R.id.nearby_layout_content, mGroupFragment).commit();
                break;
            }
        }

    }

    @Override
    public void onBackPressed() {
        if (mHeaderLayout.searchIsShowing()) {
            clearAsyncTask();
            mHeaderLayout.dismissSearch();
            mHeaderLayout.clearSearch();
            mHeaderLayout.changeSearchState(SearchState.INPUT);
        } else {
            finish();
        }
    }
}
```

17.4.4 实现“附近的群组”界面

当在顶部个人 群组中选择“群组”选项卡后，会在系统主界面中间显示“附近的群组”信息，此功能的实现布局文件是 fragment_nearbygroup.xml，具体实现代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <com.immomo.momo.android.view.MoMoRefreshExpandableList
```

```
            android:id="@+id/nearby_group_mmrelv_list"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:cacheColorHint="@color/transparent"
            android:divider="@null"
            android:fadingEdge="none"
            android:listSelector="@drawable/list_selector_transition" >
    </com.immomo.momo.android.view.MoMoRefreshExpandableList>

    <LinearLayout
        android:id="@+id/nearby_group_layout_cover"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:clickable="true" >
        <include
            layout="@layout/include_nearby_group_header"
            android:visibility="invisible" />
    </LinearLayout>
</FrameLayout>
```

“附近的群组”界面 Activity 的实现文件是 NearByGroupFragment.java，功能是在系统主界面中间加载显示附近的群组信息，并通过函数 onRefresh()进行刷新以及时显示最新群组。文件 NearByGroupFragment.java 的具体实现代码如下所示。

```
public class NearByGroupFragment extends BaseFragment implements
        OnClickListener, OnItemClickListener, OnRefreshListener,
        OnCancelListener {
    private LinearLayout mLayoutCover;
    private MoMoRefreshExpandableList mMmrelvList;
    private NearByGroupAdapter mAdapter;

    public NearByGroupFragment() {
        super();
    }

    public NearByGroupFragment(BaseApplication application, Activity activity,
            Context context) {
        super(application, activity, context);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        mView = inflater.inflate(R.layout.fragment_nearbygroup, container,
                false);
        return super.onCreateView(inflater, container, savedInstanceState);
    }

    @Override
    protected void initViews() {
        mLayoutCover = (LinearLayout) findViewById(R.id.nearby_group_layout_cover);
        mMmrelvList = (MoMoRefreshExpandableList) findViewById(R.id.nearby_group_mmrelv_list);
```

```
}

@Override
protected void initEvents() {
    mLayoutCover.setOnClickListener(this);
    mMmrelvList.setOnItemClickListener(this);
    mMmrelvList.setOnRefreshListener(this);
    mMmrelvList.setOnCancelListener(this);
}

@Override
protected void init() {
    getGroups();
}

private void getGroups() {
    if (mApplication.mNearByGroups.isEmpty()) {
        putAsyncTask(new AsyncTask<Void, Void, Boolean>() {

            @Override
            protected void onPreExecute() {
                super.onPreExecute();
                showLoadingDialog("正在加载，请稍候...");
            }

            @Override
            protected Boolean doInBackground(Void... params) {
                return JsonResolveUtils.resolveNearbyGroup(mApplication);
            }

            @Override
            protected void onPostExecute(Boolean result) {
                super.onPostExecute(result);
                dismissLoadingDialog();
                if (!result) {
                    showCustomToast("数据加载失败...");
                } else {
                    mAdapter = new NearByGroupAdapter(mApplication,
                            mContext, mApplication.mNearByGroups);
                    mMmrelvList.setAdapter(mAdapter);
                    mMmrelvList.setPinnedHeaderView(mActivity
                            .getLayoutInflater().inflate(
                                    R.layout.include_nearby_group_header,
                                    mMmrelvList, false));
                }
            }
        });
    } else {
        mAdapter = new NearByGroupAdapter(mApplication, mContext,
                mApplication.mNearByGroups);
        mMmrelvList.setAdapter(mAdapter);
```

```
                mMmrelvList.setPinnedHeaderView(mActivity.getLayoutInflater()
                        .inflate(R.layout.include_nearby_group_header, mMmrelvList,
                                false));
            }
        }

        @Override
        public void onRefresh() {
            putAsyncTask(new AsyncTask<Void, Void, Boolean>() {

                @Override
                protected Boolean doInBackground(Void... params) {
                    try {
                        Thread.sleep(2000);
                    } catch (InterruptedException e) {

                    }
                    return null;
                }

                @Override
                protected void onPostExecute(Boolean result) {
                    super.onPostExecute(result);
                    mMmrelvList.onRefreshComplete();
                }
            });

        }

        @Override
        public void onCancel() {
            clearAsyncTask();
            mMmrelvList.onRefreshComplete();
        }

        @Override
        public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {

        }

        @Override
        public void onClick(View v) {
            if (mMmrelvList.ismHeaderViewVisible()) {
                mAdapter.onPinnedHeaderClick(mMmrelvList.getFirstItemPosition());
            } else {
                mAdapter.onPinnedHeaderClick(1);
            }
        }
}
```

到此为止，本章仿陌陌系统的主要内容介绍完毕。为了节省本书篇幅，没有讲解找回密码、聊天交流、设置、留言板等信息。相关具体内容，请读者参考本书附带光盘中的源代码。

第 18 章　开发一个 Android 优化系统

在本书前面的内容中，曾经讲解过开发 Android 进程管理器的原理和演示代码。本章将通过一个综合实例的实现过程来讲解开发 Android 优化系统的基本流程。本章源码保存在“光盘：daima\18\”文件夹中。

18.1　优化大师介绍

知识点讲解：光盘：视频\视频讲解\第 18 章\优化大师介绍.avi

手机优化大师的功能是通过计算机端和手机端分别实现对 Android 手机操作系统的管理和性能优化。目前，PC 版本可以在计算机上管理手机中的通讯录、短信、应用程序和音乐等，同时通过任务管理、系统清理等功能实现对手机性能的优化。手机端版本使用了插件式的设计，可以通过设置选项中的插件安装，根据用户的喜好选择不同的功能，安装向导可以让用户选择自己所需的功能。

18.1.1　手机优化大师客户端

Android 手机优化大师的客户端，是一款运行在 Android 手机上的系统增强优化软件，目的是为用户提供便利、绿色且免费的服务，具备软件管理、任务管理、系统清理、文件管理、条码扫描等十余种功能，覆盖了日常使用的方方面面。使用手机优化大师，能够有效提高系统的整体使用效率和稳定性，帮助用户全方位管理好自己的手机。客户端的界面效果如图 18-1 所示。



图 18-1　Android 优化大师的客户端

18.1.2　手机优化大师 PC 端

手机优化大师 PC 版是一款强大的智能手机管理工具，其中 1.0 版本全面兼容基于 Android 内核的安卓

手机的管理，可以完美运行在 Windows XP/Vista 和 Win7 操作系统中，并提供如下功能。

☑ 软件商店：应用软件、游戏下载。
☑ 联系人管理：在计算机上添加、删除、修改联系人及相关归属地显示。
☑ 短信管理：在计算机上查看、发送、删除短信。
☑ 通话记录：批量删除、查看通话记录。
☑ 闹铃管理：对部分固件的 Android 手机闹铃提供了新增、删除、修改提醒的支持。
☑ 文件管理：在计算机上浏览手机 SD 卡或文件系统的内容。
☑ 书签管理：管理手机系统自带浏览器的收藏夹内容。
☑ 应用管理：在计算机上查看手机已装软件或游戏，提供批量删除版本检测等操作。
☑ 系统信息：提供较为全面的手机硬件、软件系统信息查看。
☑ 手机设置：在计算机上开关安卓设备的 Wi-Fi、蓝牙、重力感应等。
☑ 系统清理：自动扫描 Android 系统的运行临时文件和缓存。
☑ 启动管理：提供数百项软件自启动检测，轻松查找恶意软件。
☑ 屏幕截图：在计算机上截取手机屏幕，支持保存为 GIF、JPG、PNG 和 BMP 格式。
☑ 任务管理：查看手机上当前运行的应用和内存占用情况。
☑ APK 安装：内置了强大的 APK 安装器，可以检测 APK 文件中是否包含广告和安全威胁。

PC 端版本的界面效果如图 18-2 所示。



图 18-2　PC 端版本的界面效果

18.2　项 目 介 绍

知识点讲解：光盘:视频\视频讲解\第 18 章\项目介绍.avi

本项目实例的功能是，开发一个简易版的 Android 优化系统，可以实现进程维护管理和文件管理。为了

使整个项目和 18.1 中介绍的优化大师类似，还设置了其他功能，例如，手机体验、程序管理、网络管理、安装卸载、垃圾清理、节电管理和优化设置。读者可以在本实例的基础上进行扩充，实现上面的其他功能。

本项目的实现流程如图 18-3 所示。

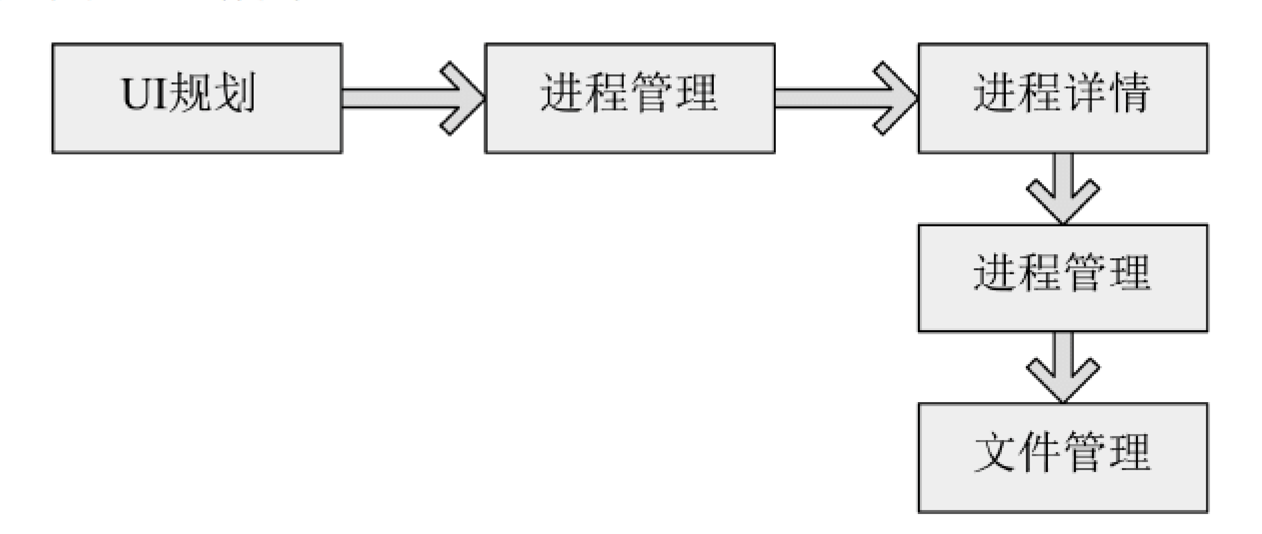


图 18-3　实现流程

18.2.1　规划 UI 界面

为了后期的升级考虑，本项目一共有 9 个模块，所以 UI 界面也包括 9 个模块主界面，UI 结构如图 18-4 所示。

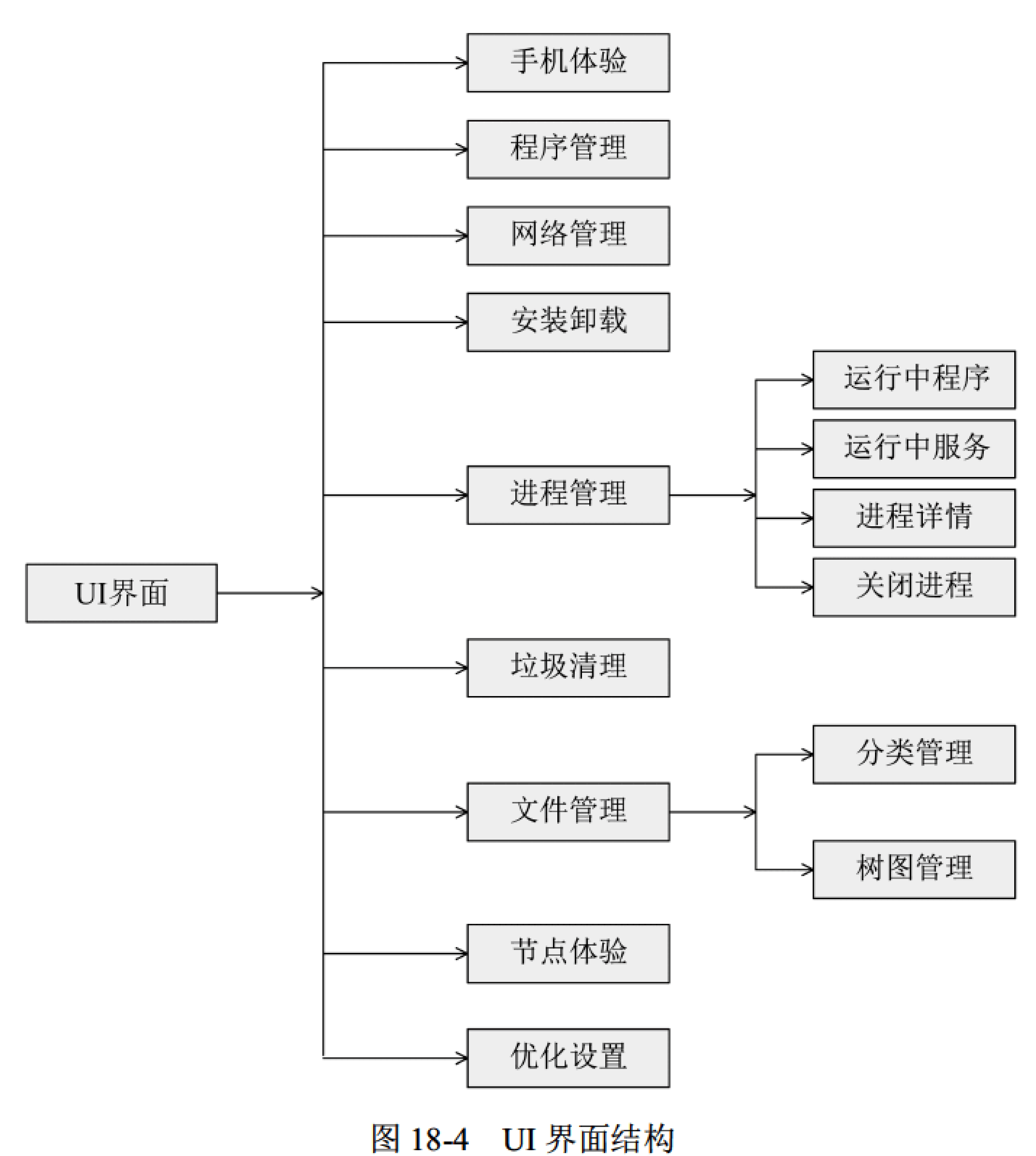


图 18-4　UI 界面结构

18.2.2　预期效果

本实例的主界面效果如图 18-5 所示。





图 18-5　主界面执行效果

18.3　准 备 工 作

知识点讲解：光盘:视频\视频讲解\第 18 章\准备工作.avi

到此为止，一个项目的准备工作就做好了。在接下来的内容中，将开始介绍本项目的具体实现过程，希望读者认真体会每一段代码的功能和编写原理，为提高自己的开发水平做准备。

18.3.1　新建工程

打开 Eclipse，依次选择 File | New | Android Project 命令，新建一个名为 AndroidManager 的工程文件，如图 18-6 所示。

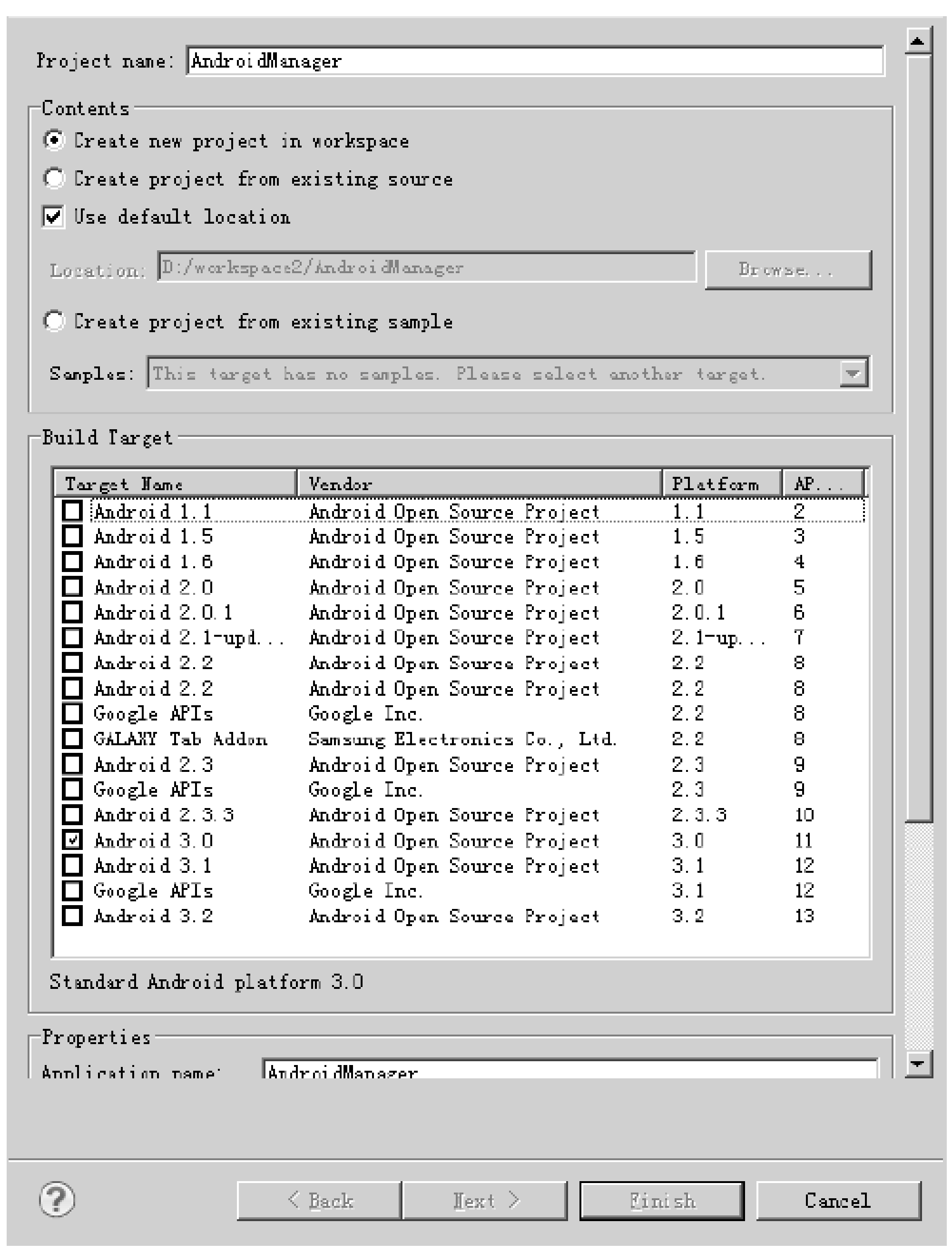


图 18-6　新建工程

18.3.2 主界面

主界面即项目执行后首先显示的界面，实现本项目主界面的流程如下所示。

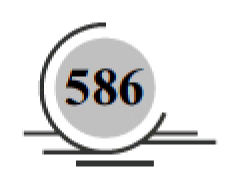
（1）编写主布局文件 main.xml，主要代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/tabhost" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout android:layout_width="fill_parent" android:orientation="vertical"
        android:layout_height="fill_parent" android:padding="5dp">
        <FrameLayout android:id="@android:id/tabcontent"
                android:padding="5dp"
                android:layout_width="fill_parent"
                android:layout_weight="1.0"
                android:layout_height="0.0dip" />
        <TabWidget android:id="@android:id/tabs"
                android:layout_width="fill_parent"
                android:visibility="gone"
                android:layout_height="wrap_content" />
        <RadioGroup android:orientation="horizontal" android:id="@+id/main_radio"
android:gravity="bottom" android:layout_width="fill_parent" android:layout_height="wrap_content"
android:layout_weight="0.0">
            <RadioButton android:id="@+id/btn1" android:layout_marginTop="2.0dip" android:tag="btn1"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:drawableTop="@drawable/process" android:text="进程" style="@style/main_tab_bottom"/>
            <RadioButton android:id="@+id/btn2" android:layout_marginTop="2.0dip" android:tag="btn2"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:drawableTop="@drawable/task" android:text="任务" style="@style/main_tab_bottom"/>
            <RadioButton android:id="@+id/btn3" android:layout_marginTop="2.0dip" android:tag="btn3"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:drawableTop="@drawable/service" android:text="服务" style="@style/main_tab_bottom"/>
            <RadioButton android:id="@+id/btn4" android:layout_marginTop="2.0dip" android:tag="btn4"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:drawableTop="@drawable/chart" android:text="图表" style="@style/main_tab_bottom"/>
            <RadioButton android:id="@+id/btn5" android:layout_marginTop="2.0dip" android:tag="btn5"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:drawableTop="@drawable/filebtn" android:text="文件" style="@style/main_tab_bottom"/>
        </RadioGroup>
    </LinearLayout>
</TabHost>
```

上述布局文件比较简单，核心功能是 RadioGroup 控件，笔者进行了最少层级优化处理。

（2）编写布局文件 nine_grid.xml，此文件的功能是实现九宫格效果，将整个界面分成 3 行 3 列的效果，具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:background="@drawable/list_bg"
    android:layout_height="fill_parent">
    <RelativeLayout android:id="@+id/toplayout" android:background="@drawable/topbg"
        android:layout_width="fill_parent" android:layout_height="wrap_content">
```



```
            <ImageView android:id="@+id/topimg" android:layout_width="wrap_content"
android:layout_marginLeft="50dp" android:layout_marginTop="8dp"
                android:layout_height="wrap_content" android:src="@drawable/glass"/>
            <TextView android:id="@+id/title" android:layout_toRightOf="@id/topimg"
android:layout_width="wrap_content" android:gravity="center" android:layout_marginLeft="10dp"
                android:text="优化大师" android:layout_height="wrap_content"
android:layout_marginTop="8dp" android:textSize="20dp"
android:textAppearance="?android:attr/textAppearanceLarge" android:textColor="#00ff00"/>
            <ImageView android:id="@+id/question" android:layout_alignParentRight="true"
android:layout_width="wrap_content" android:layout_marginRight="20dp" android:layout_marginTop="8dp"
                android:layout_height="wrap_content" android:src="@drawable/question"></ImageView>
        </RelativeLayout>
        <ImageView android:id="@+id/advertise" android:layout_width="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_alignParentLeft="true" android:layout_height="wrap_content"
            android:src="@drawable/bottomlogo" />

        <TableLayout android:clickable="true" android:focusable="true" android:layout_height="fill_parent"
android:layout_width="wrap_content" android:layout_marginTop="70dp" android:paddingLeft="30dp"
android:paddingRight="30dp">
            <TableRow android:layout_height="fill_parent" android:layout_width="wrap_content">
                <LinearLayout android:id="@+id/checkhealth"   android:layout_height="wrap_content"
android:layout_width="wrap_content" android:orientation="vertical">
                    <ImageView android:src="@drawable/checkhealth" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_alignParentTop="true"/>
                    <TextView android:text="手机体检" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_marginLeft="10dp" android:gravity="center_horizontal"/>
                </LinearLayout>
                <LinearLayout android:id="@+id/proadmin" android:layout_height="wrap_content"
android:layout_width="wrap_content" android:orientation="vertical" android:layout_marginLeft="20dp">
                    <ImageView android:src="@drawable/proadmin" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_alignParentTop="true"/>
                    <TextView android:text="程序管理" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_marginLeft="15dp" android:gravity="center_horizontal"/>
                </LinearLayout>
                <LinearLayout android:id="@+id/netadmin" android:layout_height="wrap_content"
android:layout_width="wrap_content" android:orientation="vertical" android:layout_marginLeft="20dp">
                    <ImageView android:src="@drawable/netadmin" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_alignParentTop="true"/>
                    <TextView android:text="网络管理" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_marginLeft="15dp" android:gravity="center_horizontal"/>
                </LinearLayout>
            </TableRow>
            <View android:layout_height="1dip" android:layout_width="fill_parent"
android:background="#00FF00" android:layout_marginTop="20dp" android:layout_marginBottom="20dp"/>
            <TableRow android:layout_height="fill_parent"
                android:layout_width="wrap_content">
                <LinearLayout android:id="@+id/install" android:layout_height="wrap_content"
android:layout_width="wrap_content" android:orientation="vertical">
                    <ImageView android:src="@drawable/install" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_alignParentTop="true"/>
```

```
                    <TextView android:text="安装卸载" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_marginLeft="10dp" android:gravity="center_horizontal"/>
                </LinearLayout>
                <LinearLayout android:id="@+id/adminpro" android:layout_height="wrap_content"
android:layout_width="wrap_content" android:orientation="vertical" android:layout_marginLeft="20dp">
                    <ImageView android:src="@drawable/adminpro" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_alignParentTop="true" />
                    <TextView android:text="进程管理" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_marginLeft="15dp" android:gravity="center_horizontal"/>
                </LinearLayout>
                <LinearLayout android:id="@+id/clear" android:layout_height="wrap_content"
android:layout_width="wrap_content" android:orientation="vertical" android:layout_marginLeft="20dp">
                    <ImageView android:src="@drawable/clear" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_alignParentTop="true"/>
                    <TextView android:text="垃圾清理" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_marginLeft="15dp" android:gravity="center_horizontal"/>
                </LinearLayout>
            </TableRow>
            <View android:layout_height="1dip" android:layout_width="fill_parent"
android:background="#00FF00" android:layout_marginTop="20dp" android:layout_marginBottom="20dp"/>
            <TableRow android:layout_height="fill_parent"
                android:layout_width="wrap_content">
                <LinearLayout android:id="@+id/fileadmin" android:layout_height="wrap_content"
android:layout_width="wrap_content" android:orientation="vertical">
                    <ImageView android:src="@drawable/fileadmin" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_alignParentTop="true"/>
                    <TextView android:text="文件管理" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_marginLeft="15dp" android:gravity="center_horizontal"/>
                </LinearLayout>
                <LinearLayout android:id="@+id/batteryadmin" android:layout_height="wrap_content"
android:layout_width="wrap_content" android:orientation="vertical" android:layout_marginLeft="20dp">
                    <ImageView android:src="@drawable/batteryadmin" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_alignParentTop="true"/>
                    <TextView android:text="节电管理" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_marginLeft="15dp" android:gravity="center_horizontal"/>
                </LinearLayout>
                <LinearLayout android:id="@+id/settings" android:layout_height="wrap_content"
android:layout_width="wrap_content" android:orientation="vertical" android:layout_marginLeft="20dp">
                    <ImageView android:src="@drawable/settings" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_alignParentTop="true"/>
                    <TextView android:text="优化设置" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_marginLeft="15dp" android:gravity="center_horizontal"/>
                </LinearLayout>
            </TableRow>
        </TableLayout>
</RelativeLayout>
```

在上述代码中，也进行了层级优化，布局后的界面效果如图 18-7 所示。启动 SDK 目录下的 tools 文件夹中的 hierarchyviewer.bat，可以查看当前 UI 的结构视图，如图 18-8 所示。



图 18-7　布局界面效果图

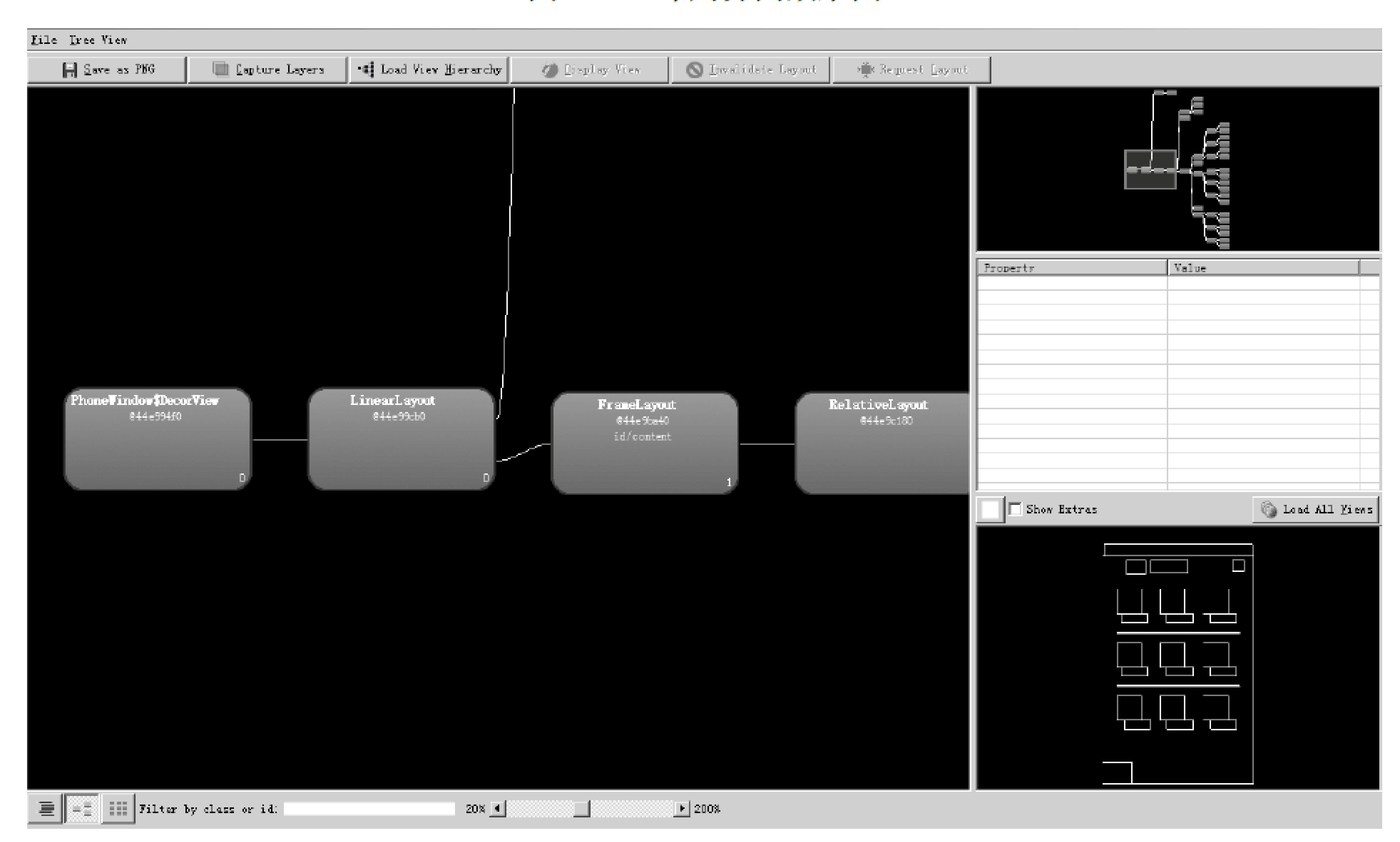


图 18-8　UI 结构视图

（3）编写文件 file_category.xml，此文件的功能是实现文件管理模块的主界面效果。具体代码如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
android:background="@drawable/list_bg" android:paddingTop="30dp" android:paddingLeft="10dp"
android:paddingRight="10dp">
    <LinearLayout android:orientation="vertical" android:layout_width="fill_parent"
android:layout_height="fill_parent">
        <LinearLayout android:orientation="horizontal" android:layout_width="fill_parent"
android:layout_height="wrap_content">
```

```
                <LinearLayout android:orientation="vertical" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_margin="20.0dip" android:layout_weight="0.5">
                    <RelativeLayout android:layout_width="fill_parent" android:layout_height="wrap_content">
                        <ImageView android:src="@drawable/file_category_pic"
android:layout_width="72.0dip" android:layout_height="72.0dip" android:baselineAlignBottom="true"
android:layout_centerHorizontal="true"/>
                    </RelativeLayout>
                    <TextView android:text="图片浏览" android:layout_width="fill_parent"
android:layout_height="wrap_content" android:gravity="center" android:textColor="@android:color/black"
android:textSize="16.0sp"/>
                </LinearLayout>
                <ImageView android:src="@drawable/main_divider" android:layout_width="2.0dip"
android:layout_height="fill_parent" android:scaleType="fitXY"/>
                <LinearLayout android:orientation="vertical" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_margin="20.0dip" android:layout_weight="0.5">
                    <RelativeLayout android:layout_width="fill_parent" android:layout_height="wrap_content">
                        <ImageView android:src="@drawable/file_category_music"
android:layout_width="72.0dip" android:layout_height="72.0dip" android:baselineAlignBottom="true"
android:layout_centerHorizontal="true"/>
                    </RelativeLayout>
                    <TextView android:text="音乐浏览" android:layout_width="fill_parent"
android:layout_height="wrap_content" android:gravity="center" android:textColor="@android:color/black"
android:textSize="16.0sp"/>
                </LinearLayout>
            </LinearLayout>
            <ImageView android:src="@drawable/main_divider" android:layout_width="fill_parent"
android:layout_height="2.0dip" android:scaleType="fitXY"/>
            <LinearLayout android:orientation="horizontal" android:layout_width="fill_parent"
android:layout_height="wrap_content">
                <LinearLayout android:orientation="vertical" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_margin="20.0dip" android:layout_weight="0.5">
                    <RelativeLayout android:layout_width="fill_parent" android:layout_height="wrap_content">
                        <ImageView android:src="@drawable/file_category_movie"
android:layout_width="72.0dip" android:layout_height="72.0dip" android:baselineAlignBottom="true"
android:layout_centerHorizontal="true"/>
                    </RelativeLayout>
                    <TextView android:text="视频浏览" android:layout_width="fill_parent"
android:layout_height="wrap_content" android:gravity="center" android:textColor="@android:color/black"
android:textSize="16.0sp"/>
                </LinearLayout>
                <ImageView android:src="@drawable/main_divider" android:layout_width="2.0dip"
android:layout_height="fill_parent" android:scaleType="fitXY"/>
                <LinearLayout android:orientation="vertical" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_margin="20.0dip" android:layout_weight="0.5">
                    <RelativeLayout android:layout_width="fill_parent" android:layout_height="wrap_content">
                        <ImageView android:src="@drawable/file_category_text"
android:layout_width="72.0dip" android:layout_height="72.0dip" android:baselineAlignBottom="true"
android:layout_centerHorizontal="true"/>
                    </RelativeLayout>
                    <TextView android:text="文档浏览" android:layout_width="fill_parent"
android:layout_height="wrap_content" android:gravity="center" android:textColor="@android:color/black"
android:textSize="16.0sp"/>
                </LinearLayout>
```



```
            </LinearLayout>
        </LinearLayout>
</RelativeLayout>
```

笔者对上述文件也专门进行了 UI 优化，读者同样可以用 hierarchyviewer.bat 查看结构视图。上述代码的 UI 效果如图 18-9 所示。



图 18-9　文件管理模块的主界面

18.4　编写主界面程序

知识点讲解：光盘:视频\视频讲解\第 18 章\编写主界面程序.avi

UI 界面设计完毕后，本节开始讲解此界面的程序文件。和此界面对应的程序文件是 NineGridActivity.java，此文件的功能是获取用户的触发事件，根据用户触摸的图标来到对应的界面。文件 NineGridActivity.java 的实现代码如下所示。

```
package com.process.ui.main;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.LinearLayout;
import com.process.R;
import com.process.ui.file.FileTabActivity;
import com.process.ui.task.TaskTabActivity;
public class NineGridActivity extends Activity implements OnClickListener {
    private LinearLayout checkhealth, proadmin, netadmin, install, adminpro,
                clear, fileadmin, batteryadmin, settings;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.nine_grid);
        setUpViews();
        setListeners();
    }
```

```
private void setUpViews() {
        checkhealth = (LinearLayout) findViewById(R.id.checkhealth);
        proadmin = (LinearLayout) findViewById(R.id.proadmin);
        netadmin = (LinearLayout) findViewById(R.id.netadmin);
        install = (LinearLayout) findViewById(R.id.install);
        adminpro = (LinearLayout) findViewById(R.id.adminpro);
        clear = (LinearLayout) findViewById(R.id.clear);
        fileadmin = (LinearLayout) findViewById(R.id.fileadmin);
        batteryadmin = (LinearLayout) findViewById(R.id.batteryadmin);
        settings = (LinearLayout) findViewById(R.id.settings);
}
private void setListeners() {
        checkhealth.setOnClickListener(this);
        proadmin.setOnClickListener(this);
        netadmin.setOnClickListener(this);
        install.setOnClickListener(this);
        adminpro.setOnClickListener(this);
        clear.setOnClickListener(this);
        fileadmin.setOnClickListener(this);
        batteryadmin.setOnClickListener(this);
        settings.setOnClickListener(this);
}
@Override
public void onClick(View v) {
        switch (v.getId()) {
        case R.id.checkhealth: {
        }
                break;
        case R.id.proadmin: {
        }
                break;
        case R.id.netadmin: {
        }
                break;
        case R.id.install: {
        }
                break;
        case R.id.adminpro: {
                Intent intent= new Intent(NineGridActivity.this,TaskTabActivity.class);
                startActivity(intent);
        }
                break;
        case R.id.clear: {
        }
                break;
        case R.id.fileadmin: {
                Intent intent= new Intent(NineGridActivity.this,FileTabActivity.class);
                startActivity(intent);
        }
                break;
        case R.id.batteryadmin: {
```

```
            }
                break;
            case R.id.settings: {
            }
                break;
            default:
                break;
            }
        }
}
```

18.5　进程管理模式模块

知识点讲解：光盘:视频\视频讲解\第 18 章\进程管理模式模块.avi

在进程管理模式中，总体设置和进程管理有关的变量，这些变量供本项目的其他模块使用。另外，还获取了每个进程所占用的内存信息和 CPU 信息，如图 18-10 所示。



图 18-10　进程列表

在本节的内容中，将详细讲解使用 Java 语言编写进程管理模式模块的具体流程。

18.5.1 基础状态文件

编写基础文件 BasicProgramUtil.java，在此文件中分别设置了图标、进程名、文件名和 CPU 模式变量，供其他模块的程序使用。文件 BasicProgramUtil.java 的实现代码如下所示。

```
package com.process.model;
import java.io.Serializable;
import android.graphics.drawable.Drawable;
public class BasicProgramUtil implements Serializable{
    /*
     * 定义应用程序的简要信息部分
     */
    private Drawable icon;                          //程序图标
    private String programName;                     //程序名称
    private String processName;
    private String cpuMemString;

    public BasicProgramUtil() {
        icon = null;
        programName = "";
        processName = "";
        cpuMemString = "";
    }
    public Drawable getIcon() {
        return icon;
    }
    public void setIcon(Drawable icon) {
        this.icon = icon;
    }
    public String getProgramName() {
        return programName;
    }
    public void setProgramName(String programName) {
        this.programName = programName;
    }
    public String getProcessName() {
        return processName;
    }
    public void setProcessName(String processName) {
        this.processName = processName;
    }
    public String getCpuMemString() {
        return cpuMemString;
    }
    public void setCpuMemString(String cpuMemString) {
        this.cpuMemString = cpuMemString;
    }
}
```

18.5.2　CPU 和内存使用信息

编写文件 CpuAndMemoryModel.java，通过此文件获取了每个进程占用的 CPU 和内存的信息，定义了和进程有关的构造函数。文件 CpuAndMemoryModel.java 的主要代码如下所示。

```
public class CpuAndMemoryModel implements Serializable {
        private String programName;
        private String processName;
        private String cpuString;
        private String memoryString;
        public String getProgramName() {
                return programName;
        }
        public void setProgramName(String programName) {
                this.programName = programName;
        }
        public String getProcessName() {
                return processName;
        }
        public void setProcessName(String processName) {
                this.processName = processName;
        }
        public String getCpuString() {
                return cpuString;
        }
        public void setCpuString(String cpuString) {
                this.cpuString = cpuString;
        }
        public String getMemoryString() {
                return memoryString;
        }
        public void setMemoryString(String memoryString) {
                this.memoryString = memoryString;
        }
}
```

18.5.3　进程详情

编写文件 DetailProgramUtil.java，功能是设置和进程有关的各个变量，显示某个进程的详细信息。文件 DetailProgramUtil.java 的主要代码如下所示。

```
public class DetailProgramUtil implements Serializable{
       private static final long serialVersionUID = 1L;
       /*
        * 定义应用程序的扩展信息部分
        */
       private int pid;
       private String processName;                                    //程序运行的进程名

       private String companyName;                                    //公司名称
       private int versionCode;                                       //版本代号
```

```
    private String versionName;                         //版本名称

    private String dataDir;                             //程序的数据目录
    private String sourceDir;                           //程序包的源目录
    private String firstInstallTime;                    //第一次安装的时间
    private String lastUpdateTime;                      //最近的更新时间

    private String userPermissions;                     //应用程序的权限
    private String activities;                          //应用程序包含的 activities
    private String services;                            //应用程序包含的服务
    //android.content.pm.PackageState 类的包信息
    //此处只是安装包的信息
    private String codeSize;
    private long dataSize;
    private long cacheSize;
    private long externalDataSize;
    private long externalCacheSize;
    private long externalMediaSize;
    private long externalObbSize;
    public DetailProgramUtil() {
        pid = 0;
        processName = "";
        companyName = "";
        versionCode = 0;
        versionName = "";
        dataDir = "";
        sourceDir = "";
        firstInstallTime = "";
        lastUpdateTime = "";
        userPermissions = "";
        activities = "";
        services = "";

        initPackageSize();
    }
    private void initPackageSize() {
        codeSize = "0.00";
        dataSize = 0;
        cacheSize = 0;
        externalCacheSize = 0;
        externalDataSize = 0;
        externalMediaSize = 0;
        externalObbSize = 0;
    }
    public int getPid() {
        return pid;
    }
    public void setPid(int pid) {
        this.pid = pid;
    }
    public int getVersionCode() {
```

```
        return versionCode;
    }
    public void setVersionCode(int versionCode) {
        this.versionCode = versionCode;
    }
    public String getVersionName() {
        return versionName;
    }
    public void setVersionName(String versionName) {
        this.versionName = versionName;
    }
    public String getCompanyName() {
        return companyName;
    }
    public void setCompanyName(String companyString) {
        this.companyName = companyString;
    }
    public String getFirstInstallTime() {
        if (firstInstallTime == null || firstInstallTime.length() <= 0) {
            firstInstallTime = "null";
        }
        return firstInstallTime;
    }
    public void setFirstInstallTime(long firstInstallTime) {
        this.firstInstallTime = DateFormat.format(
                "yyyy-MM-dd", firstInstallTime).toString();
    }
    public String getLastUpdateTime() {
        if (lastUpdateTime == null || lastUpdateTime.length() <= 0) {
            lastUpdateTime = "null";
        }
        return lastUpdateTime;
    }
    public void setLastUpdateTime(long lastUpdateTime) {
        this.lastUpdateTime = DateFormat.format(
                "yyyy-MM-dd", lastUpdateTime).toString();
    }
    public String getActivities() {
        if (activities == null || activities.length() <= 0) {
            activities = "null";
        }
        return activities;
    }
    public void setActivities(ActivityInfo[ ] activities) {
        this.activities = Array2String(activities);
    }
    public String getUserPermissions() {
        if (userPermissions == null || userPermissions.length() <= 0) {
            userPermissions = "null";
        }
        return userPermissions;
```

```
}
public void setUserPermissions(String[ ] userPermissions) {
    this.userPermissions = Array2String(userPermissions);
}
public String getServices() {
    if (services == null || services.length() <= 0) {
        services = "null";
    }
    return services;
}
public void setServices(ServiceInfo[ ] services) {
    this.services = Array2String(services);
}
public String getProcessName() {
    if (processName == null || processName.length() <= 0) {
        processName = "null";
    }
    return processName;
}
public void setProcessName(String processName) {
    this.processName = processName;
}
public String getDataDir() {
    if (dataDir == null || dataDir.length() <= 0) {
        dataDir = "null";
    }
    return dataDir;
}
public void setDataDir(String dataDir) {
    this.dataDir = dataDir;
}
public String getSourceDir() {
    if (sourceDir == null || sourceDir.length() <= 0) {
        sourceDir = "null";
    }
    return sourceDir;
}
public void setSourceDir(String sourceDir) {
    this.sourceDir = sourceDir;
}

/*
 * 3 个重载方法，参数不同，调用不同的方法，用于将 object 数组转化成要求的字符串
 */
// 用户权限信息
public String Array2String(String[ ] array) {

    String resultString = "";
    if (array != null && array.length > 0) {
        resultString = "";
        for (int i = 0; i < array.length; i++) {
```

```
                resultString += array[i];
                if (i < (array.length - 1)) {
                    resultString += "\n";
                }
            }
        }
        return resultString;
    }
    //服务信息
    public String Array2String(ServiceInfo[ ] array) {
        String resultString = "";
        if (array != null && array.length > 0) {
            resultString = "";
            for (int i = 0; i < array.length; i++) {
                if (array[i].name == null) {
                    continue;
                }
                resultString += array[i].name.toString();
                if (i < (array.length - 1)) {
                    resultString += "\n";
                }
            }
        }
        return resultString;
    }
    //活动信息
    public String Array2String(ActivityInfo[ ] array) {
        String resultString = "";
        if (array != null && array.length > 0) {
            resultString = "";
            for (int i = 0; i < array.length; i++) {
                if (array[i].name == null) {
                    continue;
                }
                resultString += array[i].name.toString();
                if (i < (array.length - 1)) {
                    resultString += "\n";
                }
            }
        }
        return resultString;
    }
    public String getCodeSize() {
        return codeSize;
    }
    public void setCodeSize(long codeSize) {
        DecimalFormat df = new DecimalFormat("###.00");
        this.codeSize = df.format((double)(codeSize/1024.0));
    }
    public long getDataSize() {
        return dataSize;
```

```
    }
    public void setDataSize(long dataSize) {
        this.dataSize = dataSize;
    }
    public long getCacheSize() {
        return cacheSize;
    }
    public void setCacheSize(long cacheSize) {
        this.cacheSize = cacheSize;
    }
    public long getExternalDataSize() {
        return externalDataSize;
    }
    public void setExternalDataSize(long externalDataSize) {
        this.externalDataSize = externalDataSize;
    }
    public long getExternalCacheSize() {
        return externalCacheSize;
    }
    public void setExternalCacheSize(long externalCacheSize) {
        this.externalCacheSize = externalCacheSize;
    }
    public long getExternalMediaSize() {
        return externalMediaSize;
    }
    public void setExternalMediaSize(long externalMediaSize) {
        this.externalMediaSize = externalMediaSize;
    }
    public long getExternalObbSize() {
        return externalObbSize;
    }
    public void setExternalObbSize(long externalObbSize) {
        this.externalObbSize = externalObbSize;
    }

    public String getPackageSize() {
        String resultString = "";
        resultString = "Code Size: " + codeSize + "KB\n"
                        + "Data Size: " + dataSize + "KB\n"
                        + "Cache Size: " + cacheSize + "KB\n"
                        + "External Data Size: " + externalDataSize + "KB\n"
                        + "External Cache Size: " + externalCacheSize + "KB\n"
                        + "External Media Size: " + externalMediaSize + "KB\n"
                        + "External Obb Size: " + externalObbSize + "KB";
        return resultString;
    }
}
```

18.6　进程视图模块

知识点讲解：光盘:视频\视频讲解\第 18 章\进程视图模块.avi

本模块的功能是在进程主界面显示当前手机的进程信息，并获取每一个进程信息对象，显示此进程的详细信息。在本节的内容中，将详细讲解使用 Java 语言编写进程视图模块的具体流程。

18.6.1　进程主视图

编写文件 MainActivity.java，功能是以列表的样式显示手机内的进程信息，分别定义了表示进程、任务、服务、图标和文件变量。文件 MainActivity.java 的主要代码如下所示。

```
public class MainActivity extends TabActivity {
	private TabHost tabHost;
	private RadioGroup mainbtGroup;
	private static final String PROCESS = "进程";
	private static final String TASK = "任务";
	private static final String SERVICE = "服务";
	private static final String CHART = "图标";
	private static final String FILE = "文件";
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.tabhost);
		tabHost = this.getTabHost();
		TabSpec tabSpec1 = tabHost.newTabSpec(PROCESS).setIndicator(PROCESS);
		tabSpec1.setContent(new Intent(this, ProcessActivity.class));
		TabSpec tabSpec2 = tabHost.newTabSpec(TASK).setIndicator(TASK);
		tabSpec2.setContent(new Intent(this, TaskActivity.class));
		TabSpec tabSpec3 = tabHost.newTabSpec(SERVICE).setIndicator(SERVICE);
		tabSpec3.setContent(new Intent(this, ServiceActivity.class));
		TabSpec tabSpec4 = tabHost.newTabSpec(CHART).setIndicator(CHART);
		tabSpec4.setContent(new Intent(this, ChartActivity.class));
		TabSpec tabSpec5 = tabHost.newTabSpec(FILE).setIndicator(FILE);
		tabSpec5.setContent(new Intent(this, FileActivity.class));

		tabHost.addTab(tabSpec1);
		tabHost.addTab(tabSpec2);
		tabHost.addTab(tabSpec3);
		tabHost.addTab(tabSpec4);
		tabHost.addTab(tabSpec5);
		mainbtGroup = (RadioGroup) this.findViewById(R.id.main_radio);
		mainbtGroup.setOnCheckedChangeListener(new OnCheckedChangeListener() {
			@Override
			public void onCheckedChanged(RadioGroup group, int checkedId) {
				switch (checkedId) {
					case R.id.btn1:
						tabHost.setCurrentTabByTag(PROCESS);
						break;
					case R.id.btn2:
```

```
                    tabHost.setCurrentTabByTag(TASK);
                    break;
                case R.id.btn3:
                    tabHost.setCurrentTabByTag(SERVICE);
                    break;
                case R.id.btn4:
                    tabHost.setCurrentTabByTag(CHART);
                    break;
                case R.id.btn5:
                    tabHost.setCurrentTabByTag(FILE);
                    break;
            }
        }
    });
  }
}
```

18.6.2 进程视图

编写文件 TaskActivity.java，此文件比较简单，功能是分别定义方法 onResume()和 oncreate()，设置进入不同的视图模式。文件 TaskActivity.java 的主要代码如下所示。

```
package com.process.ui;
import android.app.Activity;
import android.app.ListActivity;
import android.os.Bundle;
import android.util.Log;
public class TaskActivity extends ListActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d("TaskActivity", "进入 oncreate()方法");
    }

    @Override
    protected void onResume() {
        super.onResume();
        Log.d("TaskActivity", "进入 onResume()方法");
    }
}
```

18.6.3 获取进程信息

编写文件 DetailActivity.java，功能是根据某个进程的名字获取应用程序的 ApplicationInfo 对象，然后显示出此进程的详细信息。文件 DetailActivity.java 的主要代码如下所示。

```
public class DetailActivity extends Activity {
    private PackageManager packageManager;
    private ProcessMemoryUtil processMemoryUtil;
    private PackageUtil packageUtil;
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        packageUtil = new PackageUtil(DetailActivity.this);
        Intent intent = getIntent();
        Bundle bundle = intent.getExtras();
        String procNameString = bundle.getString("procNameString");
        TextView tv = new TextView(DetailActivity.this);
        tv.setText(procNameString);
        setContentView(tv);
    }
    public DetailProgramUtil buildProgramUtilComplexInfo(String procNameString) {
        DetailProgramUtil complexProgramUtil = new DetailProgramUtil();
        //根据进程名，获取应用程序的 ApplicationInfo 对象
        ApplicationInfo tempAppInfo = packageUtil.getApplicationInfo(procNameString);
        if (tempAppInfo == null) {
            return null;
        }
        PackageInfo tempPkgInfo = null;
        try {
            tempPkgInfo = packageManager.getPackageInfo(
                    tempAppInfo.packageName,
                    PackageManager.GET_UNINSTALLED_PACKAGES |
                    PackageManager.GET_ACTIVITIES
                    | PackageManager.GET_SERVICES | PackageManager.GET_PERMISSIONS);
        } catch (NameNotFoundException e) {
            e.printStackTrace();
        }
        if (tempPkgInfo == null) {
            return null;
        }
        complexProgramUtil.setProcessName(procNameString);
        complexProgramUtil.setCompanyName(getString(R.string.no_use));
        complexProgramUtil.setVersionName(tempPkgInfo.versionName);
        complexProgramUtil.setVersionCode(tempPkgInfo.versionCode);
        complexProgramUtil.setDataDir(tempAppInfo.dataDir);
        complexProgramUtil.setSourceDir(tempAppInfo.sourceDir);
        //以下注释部分的信息暂时获取不到
//      complexProgramUtil.setFirstInstallTime(tempPkgInfo.firstInstallTime);
//      complexProgramUtil.setLastUpdateTime(tempPkgInfo.lastUpdateTime);
//      complexProgramUtil.setCodeSize(packageStats.codeSize);
//      complexProgramUtil.setDataSize(packageStats.dataSize);
//      complexProgramUtil.setCacheSize(packageStats.cacheSize);
//      complexProgramUtil.setExternalDataSize(0);
//      complexProgramUtil.setExternalCacheSize(0);
//      complexProgramUtil.setExternalMediaSize(0);
//      complexProgramUtil.setExternalObbSize(0);
        //获取以下 3 个信息，需要为 PackageManager 进行授权(packageManager.getPackageInfo()方法)
        complexProgramUtil.setUserPermissions(tempPkgInfo.requestedPermissions);
        complexProgramUtil.setServices(tempPkgInfo.services);
        complexProgramUtil.setActivities(tempPkgInfo.activities);
        return complexProgramUtil;
```

```
    }
}
```

18.7　进程类别模块

知识点讲解：光盘:视频\视频讲解\第 18 章\进程类别模块.avi

本模块的功能是将进程分为两类：运行程序和运行中服务，并且设置了进度条效果显示系统内的进程。在本节的内容中，将详细讲解使用 Java 语言编写进程类别模块的具体流程。

18.7.1　加载进程

编写文件 ProcessActivity.java，功能是以进度条的样式加载当前手机中运行的进程。文件 ProcessActivity.java 的主要代码如下所示。

```
public class ProcessActivity extends ListActivity implements OnItemLongClickListener,OnItemClickListener{
    private PackageManager packageManager;
    private ProgressDialog pd;
    private Handler handler;
    private List<BasicProgramUtil> list = null;
    private PackageUtil packageUtil;
    private ProcessMemoryUtil processMemoryUtil;
    private ListView listView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.process);
        listView = getListView();
        listView.setOnItemLongClickListener(this);//为 listView 添加
        listView.setOnItemClickListener(this);
        packageUtil = new PackageUtil(ProcessActivity.this);
        processMemoryUtil = new ProcessMemoryUtil();
        packageManager = getPackageManager();
        pd = new ProgressDialog(ProcessActivity.this);//生成一个进度条
        pd.setProgressStyle(ProgressDialog.STYLE_SPINNER);
        pd.setTitle(getString(R.string.progress_tips_title));
        pd.setMessage(getString(R.string.progress_tips_content));
        handler = new RefreshHandler();
        pd.show();
        RefreshThread thread = new RefreshThread();
        thread.start();//耗时操作，需要开启一个线程
    }

    @Override
    protected void onResume() {
        super.onResume();
    }
    class RefreshHandler extends Handler {
        @Override
```

```
            public void handleMessage(Message msg) {
                  refreshListItems();
                  setTitle("软件信息,有" + list.size() + "个进程在运行.");
                  pd.dismiss();//关闭进度条
            }
      }
      class RefreshThread extends Thread {
            @Override
            public void run() {
                  getRunningAppProcesses();
                  Message msg = handler.obtainMessage();
                  handler.sendMessage(msg);
            }
      }
      private void refreshListItems() {
            list = getRunningAppProcesses();
            MyAdapter adapter = new MyAdapter(ProcessActivity.this, list);
            listView.setAdapter(adapter);
      }

      private List<BasicProgramUtil> getRunningAppProcesses() {
            ActivityManager activityManager = (ActivityManager) getSystemService(ACTIVITY_SERVICE);
            List<RunningAppProcessInfo> procList = activityManager.getRunningAppProcesses();
            List list = new ArrayList<BasicProgramUtil>();
            for (Iterator<RunningAppProcessInfo> iterator = procList.iterator(); iterator
                        .hasNext();) {
                  RunningAppProcessInfo procInfo = iterator.next();
                  BasicProgramUtil basicProgramUtil = buildProgramUtilSimpleInfo(procInfo.pid, procInfo.
processName);
                  list.add(basicProgramUtil);
            }
            return list;
      }
      private void returnToHome() {
          Intent home = new Intent(Intent.ACTION_MAIN);
          home.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
          home.addCategory(Intent.CATEGORY_HOME);
          startActivity(home);
     }
      public BasicProgramUtil buildProgramUtilSimpleInfo(int procId, String procNameString) {
          BasicProgramUtil programUtil = new BasicProgramUtil();
          programUtil.setProcessName(procNameString);
          //根据进程名获取应用程序的 ApplicationInfo 对象
          ApplicationInfo tempAppInfo = packageUtil.getApplicationInfo(procNameString);
          if (tempAppInfo != null) {
               //为进程加载图标和程序名称
               programUtil.setIcon(tempAppInfo.loadIcon(packageManager));
               programUtil.setProgramName(tempAppInfo.loadLabel(packageManager).toString());
          }
          else {
               //如果获取失败，则使用默认的图标和程序名
```

```
programUtil.setIcon(getApplicationContext().getResources().getDrawable(R.drawable.unknown));
                programUtil.setProgramName(procNameString);
            }
            String str = processMemoryUtil.getMemInfoByPid(procId);
            programUtil.setCpuMemString(str);
            return programUtil;
        }
    class MyAdapter extends BaseAdapter{
            private Context context;
            private List<BasicProgramUtil> list;
            private LayoutInflater inflater;
            public MyAdapter(Context context,List<BasicProgramUtil> list){
                super();
                this.context = context;
                this.list = list;
                this.inflater = LayoutInflater.from(context);
            }
            @Override
            public int getCount() {
                return list.size();
            }
            @Override
            public Object getItem(int position) {
                return list.get(position);
            }
            @Override
            public long getItemId(int position) {
                return position;
            }
            @Override
            public View getView(int position, View convertView, ViewGroup parent) {
                BasicProgramUtil bp = list.get(position);
                View v = convertView;
                if(v==null){
                    v = inflater.inflate(R.layout.proc_list_item, null);
                    ViewHolder viewHolder = new ViewHolder();
                    viewHolder.img = (ImageView)v.findViewById(R.id.icon);
                    viewHolder.tv1 =  (TextView)v.findViewById(R.id.programName);
                    //viewHolder.tv2 =  (TextView)v.findViewById(R.id.processName);
                    viewHolder.tv3 =  (TextView)v.findViewById(R.id.cpuMemString);
                    v.setTag(viewHolder);
                }
                ViewHolder viewHolder = (ViewHolder)v.getTag();
                viewHolder.img.setBackgroundDrawable(bp.getIcon());
                viewHolder.tv1.setText(bp.getProgramName());
                //viewHolder.tv2.setText(bp.getProcessName());
                viewHolder.tv3.setText(bp.getCpuMemString());
                return v;
            }
    }
    static class ViewHolder{
```

```
        private ImageView img;
        private TextView tv1;
        //private TextView tv2;
        private TextView tv3;
    }
    @Override
    public boolean onItemLongClick(AdapterView<?> arg0, View arg1, int position,long arg3) {
        return false;
    }
    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int position, long arg3) {
        final BasicProgramUtil bsu = list.get(position);
        final Intent intent = new Intent(ProcessActivity.this,DetailActivity.class);
        Bundle bundle = new Bundle();
        bundle.putString("procNameString", bsu.getProcessName());
        intent.putExtras(bundle);

        AlertDialog.Builder builder = new AlertDialog.Builder(ProcessActivity.this);
        builder.setTitle("查看详情 or 结束此进程");
        builder.setIcon(R.drawable.question);
        builder.setPositiveButton("详情", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                startActivity(intent);//跳往程序详情显示页面
            }
        });
        builder.setNegativeButton("结束此进程", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                //结束此进程
            }
        });
        builder.show();
    }
}
```

18.7.2　后台加载设置

编写文件 ServiceActivity.java，功能是根据用户的选择加载执行不同的方法，这样可以分别进入运行中程序和运行中服务模式。文件 ServiceActivity.java 的主要代码如下所示。

```
public class ServiceActivity extends ListActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d("ServiceActivity", "进入 onCreate()方法");
    }
    @Override
    protected void onResume() {
        super.onResume();
        Log.d("ServiceActivity", "进入 onResume()方法");
```

```
    }
}
```

18.7.3 加载显示

编写文件 TaskTabActivity.java，功能是分别定义两个 View 对象 view1 和 view2，根据用户的需要进入不同的视图界面。其中，view1 表示运行中的程序视图，view2 表示运行中的服务视图。文件 TaskTabActivity.java 的主要代码如下所示。

```
public class TaskTabActivity extends TabActivity {
    private TabHost tabHost;
    private static final String RUNNINGPROGRAM = "运行中程序";
    private static final String RUNNINGSERVICE = "运行中服务";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tabhost);
        tabHost = getTabHost();
        View view1 = View.inflate(TaskTabActivity.this, R.layout.tab, null);
        ((ImageView)
view1.findViewById(R.id.tab_imageview_icon)).setImageResource(R.drawable.task_tab1_icon);
        ((TextView) view1.findViewById(R.id.tab_textview_title)).setText(RUNNINGPROGRAM);
        TabHost.TabSpec spec1 = tabHost.newTabSpec(RUNNINGPROGRAM)
                .setIndicator(view1)
                .setContent(new Intent(this, ProcessActivity.class));
        tabHost.addTab(spec1);
        View view2 = View.inflate(TaskTabActivity.this, R.layout.tab, null);
        ((ImageView)
view2.findViewById(R.id.tab_imageview_icon)).setImageResource(R.drawable.task_tab2_icon);
        ((TextView) view2.findViewById(R.id.tab_textview_title)).setText(RUNNINGSERVICE);
        TabHost.TabSpec spec2 = tabHost.newTabSpec(RUNNINGPROGRAM)
                .setIndicator(view2)
                .setContent(new Intent(this, ServiceActivity.class));
        tabHost.addTab(spec2);
    }
}
```

18.8 文件管理模式模块

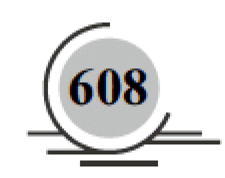
知识点讲解：光盘:视频\视频讲解\第 18 章\文件管理模式模块.avi

本模块的功能是设置当前手机设备中的文件模式，可以将文件分为不同的类别，并快速打开相应类别的文件。在本节的内容中，将详细讲解使用 Java 语言编写文件管理模式模块的具体流程。

18.8.1 文件分类

编写文件 FileCategoryActivity.java，功能是实现文件的分类，将当前手机设备中的文件分为如下 4 种类型。

☑ 图片。



☑ 音乐。
☑ 视频。
☑ 文档。

文件 FileCategoryActivity.java 的实现代码如下所示。

```
package com.process.ui.file;
import com.process.R;
import android.app.Activity;
import android.os.Bundle;
public class FileCategoryActivity extends Activity {
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.file_category);
	}
}
```

18.8.2　加载进程

编写文件 FileActivity.java，功能是响应用户的选择，并根据选择显示左侧或右侧对应目录中的子目录。文件 FileActivity.java 的主要代码如下所示。

```
public class FileActivity extends Activity{
	private ListView leftLV,rightLV;
	List<Map<String, Object>> leftList;
	List<Map<String, Object>> rightList;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.file);

		leftLV = (ListView)findViewById(R.id.leftLV);
		rightLV = (ListView)findViewById(R.id.rightLV);

		List<Map<String, Object>> fileList = new ArrayList<Map<String,Object>>();
		FileUtil.getParentPath(new File("/"), fileList);
		leftList = fileList;
		rightList = FileUtil.getSubDirAndFiles(new File("/"));

		setUpAdapter();//填充初始数据
		leftLV.setOnItemClickListener(new LeftItemListener());
		rightLV.setOnItemClickListener(new RightItemListener());
		rightLV.setOnItemLongClickListener(new rightLVItemLongClickListener());
	}
	private void setUpAdapter(){
		if(leftList!=null){
			SimpleAdapter leftAdapter = new SimpleAdapter(this, leftList, R.layout.file_left_item,
					new String[ ] { "currentDirImage", "currentDirName"}, new int[ ] { R.id.currentDirImage,
							R.id.currentDirName});
			leftLV.setAdapter(leftAdapter);
		}else{
```

```
                    leftLV.setAdapter(null);
                }
                if(rightList!=null){
                    SimpleAdapter rightAdapter = new SimpleAdapter(this, rightList, R.layout.file_right_item,
                            new String[ ] { "subDirImage", "subDirName"}, new int[ ] { R.id.subDirImage,
                                    R.id.subDirName});
                    rightLV.setAdapter(rightAdapter);
                }else{
                    rightLV.setAdapter(null);
                    Toast.makeText(FileActivity.this, "空文件夹", Toast.LENGTH_SHORT).show();
                }
            }
            class LeftItemListener implements OnItemClickListener{
                @Override
                public void onItemClick(AdapterView<?> arg0, View arg1, int position,
                        long arg3) {
                    Map<String, Object> map = leftList.get(position);
                    String currentDirPath = (String)map.get("currentDirPath");
                    File file = new File(currentDirPath);

                    List<Map<String, Object>> list = new ArrayList<Map<String, Object>>();
                    FileUtil.getParentPath(file, list);
                    leftList = list;
                    rightList = FileUtil.getSubDirAndFiles(file);
                    setUpAdapter();//刷新
                }
            }
            class RightItemListener implements OnItemClickListener{
                @Override
                public void onItemClick(AdapterView<?> arg0, View arg1, int position,
                        long arg3) {
                    Map<String, Object> map = rightList.get(position);
                    String subDirPath = (String)map.get("subDirPath");
                    File file = new File(subDirPath);
                    File parentFile = file.getParentFile();

                    if(file.isDirectory()){//处理左边目录与右边目录以及右边文件的显示
                        List<Map<String, Object>> list = new ArrayList<Map<String, Object>>();
                        FileUtil.getParentPath(file, list);
                        leftList = list;
                        rightList = FileUtil.getSubDirAndFiles(file);
                        setUpAdapter();//刷新

                    }else{//如果单击的是文件，提示用户选择相应的程序打开此文件
                        Toast.makeText(FileActivity.this, "你选择的是文件", Toast.LENGTH_SHORT).show();
                    }
                }
            }
            @Override
            public boolean onCreateOptionsMenu(Menu menu){
                super.onCreateOptionsMenu(menu);
                MenuInflater menuInflater = getMenuInflater();
                menuInflater.inflate(R.menu.filemenu, menu);
```

```
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case R.id.addfolder:{
                        AlertDialog.Builder builder = new AlertDialog.Builder(FileActivity.this);
                        builder.setTitle("输入名字");
                        builder.setIcon(R.drawable.directory);
                        builder.setCancelable(true);

                        LayoutInflater inflater = LayoutInflater.from(FileActivity.this);
                        View rootView = inflater.inflate(R.layout.input_foldername_dialog, null);
                        final EditText et = (EditText)rootView.findViewById(R.id.foldername);

                        builder.setView(rootView);

                        builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                String foldername = et.getText().toString();
                                Map<String, Object> map = leftList.get(0);

                                File parentFile = new File((String)map.get("currentDirPath"));
                                File newFolder = new File(parentFile,foldername);
                                if(newFolder.mkdir()){
                                    rightList = FileUtil.getSubDirAndFiles(parentFile);
                                    setUpAdapter();//刷新
                                    Toast.makeText(FileActivity.this, "创建成功",
                                    Toast.LENGTH_SHORT).show();
                                }else{
                                    Toast.makeText(FileActivity.this, "重名文件夹",
                                    Toast.LENGTH_SHORT).show();
                                }
                            }
                        });
                        builder.show();
                    }
            return true;
        case R.id.deletefolder:{
                        Toast.makeText(FileActivity.this, "删除文件夹",
                                    Toast.LENGTH_SHORT).show();
                    }
            return true;
        default:
            return false;
        }
    }
    class rightLVItemLongClickListener implements OnItemLongClickListener{
        @Override
        public boolean onItemLongClick(AdapterView<?> arg0, View arg1,
                    int position, long arg3) {
            final Map<String, Object> map = rightList.get(0);
```

```
                    AlertDialog.Builder builder = new AlertDialog.Builder(FileActivity.this);
                    builder.setTitle("你确定要删除吗?");
                    builder.setIcon(R.drawable.question);
                    builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            File currentFile = new File((String)map.get("subDirPath"));
                            if(currentFile.delete()){
                                rightList = FileUtil.getSubDirAndFiles(currentFile.getParentFile());
                                setUpAdapter();//刷新
                                Toast.makeText(FileActivity.this, "删除成功", Toast.LENGTH_SHORT).show();
                            }else{
                                Toast.makeText(FileActivity.this, "删除失败", Toast.LENGTH_SHORT).show();
                            }
                        }
                    });
                    builder.setNegativeButton("取消", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                        }
                    });
                    builder.show();
                    return false;
            }

        }
}
```

18.8.3 文件视图处理

除了将文件按照类别进行管理外，还可以根据树图模式进行管理。编写文件 FileTabActivity.java，根据用户选择的选项卡显示不同的文件管理模式，此文件的主要代码如下所示。

```
public class FileTabActivity extends TabActivity {
    private TabHost tabHost;
    private static final String VIEWBYTYPE = "分类管理";
    private static final String TREEADMIN = "树图管理";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.tabhost);
        tabHost = getTabHost();
        View view1 = View.inflate(FileTabActivity.this, R.layout.tab, null);
        ((ImageView)
view1.findViewById(R.id.tab_imageview_icon)).setImageResource(R.drawable.file_tab1_icon);
        ((TextView) view1.findViewById(R.id.tab_textview_title)).setText(VIEWBYTYPE);
        TabHost.TabSpec spec1 = tabHost.newTabSpec(VIEWBYTYPE)
                .setIndicator(view1)
                .setContent(new Intent(this, FileCategoryActivity.class));
        tabHost.addTab(spec1);
        View view2 = View.inflate(FileTabActivity.this, R.layout.tab, null);
        ((ImageView)
view2.findViewById(R.id.tab_imageview_icon)).setImageResource(R.drawable.file_tab2_icon);
```

```
            ((TextView) view2.findViewById(R.id.tab_textview_title)).setText(TREEADMIN);
            TabHost.TabSpec spec2 = tabHost.newTabSpec(VIEWBYTYPE)
                        .setIndicator(view2)
                        .setContent(new Intent(this, FileActivity.class));
            tabHost.addTab(spec2);
      }
}
```

如果用户选择“分类管理”选项卡，则显示如图 18-9 所示的界面，如果选择“树图管理”选项卡，则显示如图 18-11 所示的界面。

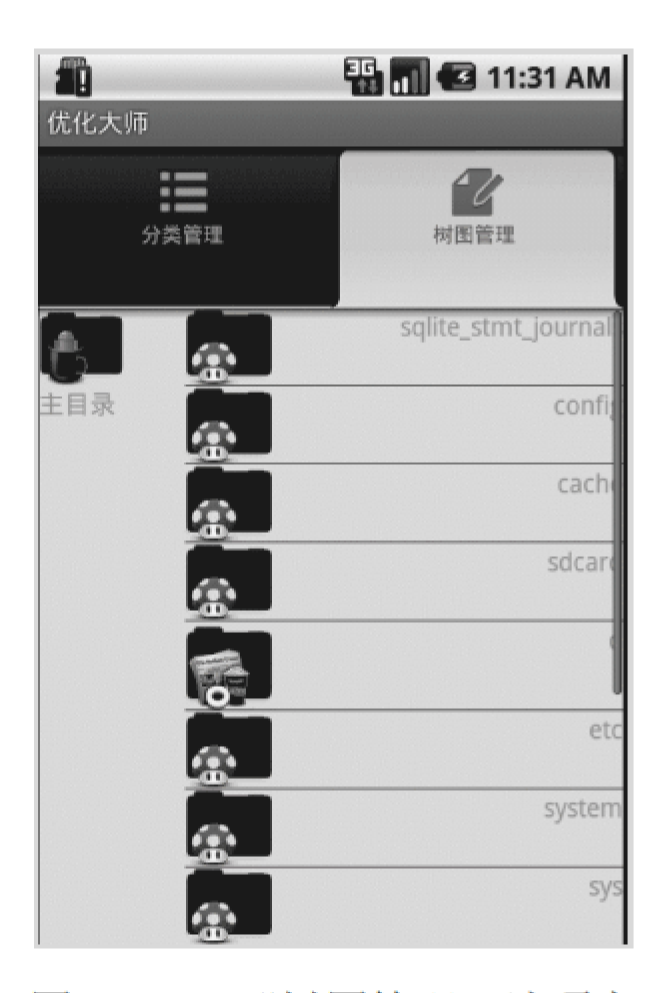


图 18-11　“树图管理”选项卡

18.9　文件管理模块

知识点讲解：光盘:视频\视频讲解\第 18 章\文件管理模块.avi

本模块的功能是管理当前手机设备中的文件，查看某个目录下的子目录信息，并且对内存和 CPU 的使用信息进行了转换处理。在本节的内容中，将详细讲解使用 Java 语言编写文件管理模块的具体流程。

18.9.1　文件夹

编写文件 PackageUtil.java，功能是通过包管理器检索所有的应用程序（包括卸载）与数据目录。此文件的主要代码如下所示。

```
public class PackageUtil {
    //ApplicationInfo 类，保存了普通应用程序的信息，主要是指 Manifest.xml 中 application 标签中的信息
    private List<ApplicationInfo> allAppList;
    public PackageUtil(Context context) {
        //通过包管理器，检索所有的应用程序（包括卸载）与数据目录
        PackageManager pm = context.getApplicationContext().getPackageManager();
        allAppList = pm.getInstalledApplications(PackageManager.GET_UNINSTALLED_PACKAGES);
        pm.getInstalledPackages(0);
    }
    /**
```

```
        * 通过一个程序名返回该程序的一个 ApplicationInfo 对象
        * @param name  程序名
        * @return ApplicationInfo
        */
    public ApplicationInfo getApplicationInfo(String appName) {
        if (appName == null) {
            return null;
        }
        for (ApplicationInfo appinfo : allAppList) {
            if (appName.equals(appinfo.processName)) {
                return appinfo;
            }
        }
        return null;
    }
}
```

18.9.2 显示文件信息

编写文件 FileUtil.java，其功能是获取并显示当前手机设备中的文件信息，包括文件名、文件格式和文件路径。文件 FileUtil.java 的主要代码如下所示。

```
public class FileUtil {
    public static void getParentPath(File file,List<Map<String,Object>> list){
        Map<String,Object> map = new HashMap<String, Object>();
    if(file.getName()==null||"".equals(file.getName())||"/".equals(file.getName())){
            map.put("currentDirName","主目录");
            map.put("currentDirImage",R.drawable.rootdir);
        }else if(file.getName().indexOf("sdcard")!=-1){
            map.put("currentDirName","sdcard");
            map.put("currentDirImage",R.drawable.sdcard);
        }else{
            map.put("currentDirName", file.getName());
            map.put("currentDirImage",R.drawable.directory);
        }
        map.put("currentDirPath", file.getAbsolutePath());
        list.add(map);
        if(file.getParentFile()!=null){
            getParentPath(file.getParentFile(),list);
        }
    }
    public static List<Map<String, Object>> getSubDirAndFiles(File pathFile){
        File[ ] files = pathFile.listFiles();
        if(files==null||files.length<1){
            return null;
        }
        List<Map<String, Object>> list = new ArrayList<Map<String, Object>>(files.length);
        for (File file : files){
            Map<String, Object> map = new HashMap<String, Object>();
            if(file.isDirectory()){
                map.put("subDirImage", R.drawable.directory);
```

```
                }else{
                    String fileName = file.getName();
                    if(fileName.indexOf("jpg")!=-1){
                        map.put("subDirImage", R.drawable.jpg);
                    }else if(fileName.indexOf("txt")!=-1){
                        map.put("subDirImage", R.drawable.txt);
                    }else if(fileName.indexOf("mp3")!=-1){
                        map.put("subDirImage", R.drawable.mp3);
                    }else if(fileName.indexOf("avi")!=-1){
                        map.put("subDirImage", R.drawable.avi);
                    }else if(fileName.indexOf("xls")!=-1){
                        map.put("subDirImage", R.drawable.excel);
                    }else if(fileName.indexOf("mpeg")!=-1){
                        map.put("subDirImage", R.drawable.mpeg);
                    }else if(fileName.indexOf("rar")!=-1){
                        map.put("subDirImage", R.drawable.rar);
                    }else if(fileName.indexOf("tif")!=-1){
                        map.put("subDirImage", R.drawable.tif);
                    }else if(fileName.indexOf("wav")!=-1){
                        map.put("subDirImage", R.drawable.wav);
                    }else if(fileName.indexOf("wma")!=-1){
                        map.put("subDirImage", R.drawable.wma);
                    }else if(fileName.indexOf("doc")!=-1){
                        map.put("subDirImage", R.drawable.word);
                    }else if(fileName.indexOf("zip")!=-1){
                        map.put("subDirImage", R.drawable.zip);
                    }else{
                        map.put("subDirImage", R.drawable.file);
                    }
                }
                map.put("subDirName", file.getName());
                map.put("subDirPath", file.getPath());
                list.add(map);
            }
            return list;
        }
}
```

18.9.3　操作文件

编写文件 CMDExecute.java，功能是定义 ProcessBuilder 对象 builder，通过 CMD 方式操作一个文件。如果文件的路径为空，则关闭操作流。文件 CMDExecute.java 的主要代码如下所示。

```
public class CMDExecute {
    public synchronized String run(String[ ] cmd, String workdirectory)
                throws IOException {
        String result = "";
        try {
            ProcessBuilder builder = new ProcessBuilder(cmd);
            InputStream in = null;
            //设置一个路径
```

```
                        if (workdirectory != null) {
                                builder.directory(new File(workdirectory));
                                builder.redirectErrorStream(true);
                                Process process = builder.start();
                                in = process.getInputStream();
                                byte[ ] re = new byte[1024];
                                while (in.read(re) != -1)
                                        result = result + new String(re);
                        }
                        if (in != null) {
                                in.close();
                        }
                } catch (Exception ex) {
                        ex.printStackTrace();
                }
                return result;
        }
}
```

18.9.4 获取进程的 CPU 和内存信息

编写文件 ProcessMemoryUtil.java，获取指定进程所占用的 CPU 信息和内存信息。在获取 CPU 的使用信息时，进行了百分比计算。文件 ProcessMemoryUtil.java 的主要实现代码如下所示。

```
public class ProcessMemoryUtil {

        private static final int INDEX_FIRST = -1;
        private static final int INDEX_PID = INDEX_FIRST + 1;
        private static final int INDEX_CPU = INDEX_FIRST + 2;
        private static final int INDEX_STAT = INDEX_FIRST + 3;
        private static final int INDEX_THR = INDEX_FIRST + 4;
        private static final int INDEX_VSS = INDEX_FIRST + 5;
        private static final int INDEX_RSS = INDEX_FIRST + 6;
        private static final int INDEX_PCY = INDEX_FIRST + 7;
        private static final int INDEX_UID = INDEX_FIRST + 8;
        private static final int INDEX_NAME = INDEX_FIRST + 9;
        private static final int Length_ProcStat = 9;

        private List<String[ ]> PMUList = null;

        public ProcessMemoryUtil() {
         initPMUtil();
        }

        private String getProcessRunningInfo() {
                Log.i("fetch_process_info", "start... ");
                String result = null;
                CMDExecute cmdexe = new CMDExecute();
                try {
                        String[ ] args = {"/system/bin/top", "-n", "1"};
```

```
            result = cmdexe.run(args, "/system/bin/");
        } catch (IOException ex) {
            Log.i("fetch_process_info", "ex=" + ex.toString());
        }
        return result;
    }

    private int parseProcessRunningInfo(String infoString) {
        String tempString = "";
        boolean bIsProcInfo = false;

        String[ ] rows = null;
        String[ ] columns = null;
        rows = infoString.split("[\n]+");          //使用正则表达式分割字符串

        for (int i = 0; i < rows.length; i++) {
            tempString = rows[i];
            if (tempString.indexOf("PID") == -1) {
                if (bIsProcInfo == true) {
                    tempString = tempString.trim();
                    columns = tempString.split("[ ]+");
                    if (columns.length == Length_ProcStat) {
                        PMUList.add(columns);
                    }
                }
            } else {
                bIsProcInfo = true;
            }
        }

        return PMUList.size();
    }

    //初始化所有进程的 CPU 和内存列表，用于检索每个进程的信息
    public void initPMUtil() {
        PMUList = new ArrayList<String[ ]>();
        String resultString = getProcessRunningInfo();
        parseProcessRunningInfo(resultString);
    }

    //根据进程名获取 CPU 和内存信息
    public String getMemInfoByName(String procName) {
        String result = "";

        String tempString = "";
        for (Iterator<String[ ]> iterator = PMUList.iterator(); iterator.hasNext();) {
            String[ ] item = (String[ ]) iterator.next();
            tempString = item[INDEX_NAME];
            if (tempString != null && tempString.equals(procName)) {
```

```
                result = "CPU:" + item[INDEX_CPU]
                        + "内存:" + item[INDEX_RSS];
                break;
            }
        }
        return result;
    }

    //根据进程 ID 获取 CPU 和内存信息
    public String getMemInfoByPid(int pid) {
        String result = "";

        String tempPidString = "";
        int tempPid = 0;
        int count = PMUList.size();
        for (int i = 0; i < count; i++) {
            String[ ] item = PMUList.get(i);
            tempPidString = item[INDEX_PID];
            if (tempPidString == null) {
                continue;
            }
            tempPid = Integer.parseInt(tempPidString);
            if (tempPid == pid) {
                result = "CPU:" + item[INDEX_CPU]
                        + "内存:" + item[INDEX_RSS];
                break;
            }
        }
        return result;
    }

    //根据进程 ID 获取内存信息
    public String getMemorySizeByPid(int pid) {
        String result = "";

        String tempPidString = "";
        int tempPid = 0;
        int count = PMUList.size();
        for (int i = 0; i < count; i++) {
            String[ ] item = PMUList.get(i);
            tempPidString = item[INDEX_PID];
            if (tempPidString == null) {
                continue;
            }
            tempPid = Integer.parseInt(tempPidString);
            if (tempPid == pid) {
              int size = item[INDEX_RSS].length();
                result = item[INDEX_RSS].substring(0,size-1);
                break;
            }
        }
        return result;
```

```
    }

    // 根据进程 ID 获取 CPU 信息
    public String getCPUSizeByPid(int pid) {
        String result = "";
        String tempPidString = "";
        int tempPid = 0;
        int count = PMUList.size();
        for (int i = 0; i < count; i++) {
            String[ ] item = PMUList.get(i);
            tempPidString = item[INDEX_PID];
            if (tempPidString == null) {
                continue;
            }
            tempPid = Integer.parseInt(tempPidString);
            if (tempPid == pid) {
                result = item[INDEX_CPU];
                break;
            }
        }
        return result;
    }
}
```

18.10　系 统 测 试

知识点讲解：光盘:视频\视频讲解\第 18 章\系统测试.avi

经过本章前面内容的讲解，一个基本的简化版 Android 优化系统开发完毕。本节将开始进行具体测试。单击图 18-12 中的“进程管理”按钮来到进程管理界面，如图 18-13 所示。



图 18-12　主界面执行效果



图 18-13　进程管理界面

选中图 18-13 中的某个进程后会弹出一个提示框，如图 18-14 所示。

单击“详情”按钮可以在新界面中显示此进程的详细信息，如图 18-15 所示。单击“结束此进程”按钮后可以关闭进程。



图 18-14 提示框

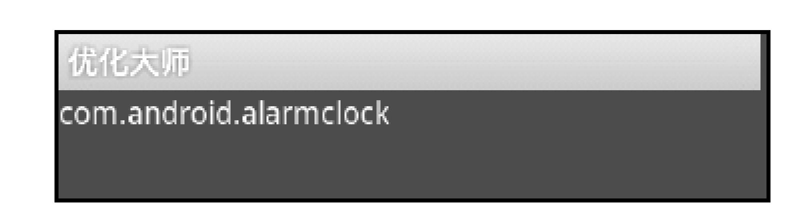


图 18-15 某进程的详情

选择图 18-13 中顶部的“运行中服务”选项卡可以来到一个新界面，如图 18-16 所示。

单击图 18-12 中的“文件管理”按钮来到文件管理界面，如图 18-17 所示。



图 18-16 “运行中服务”界面



图 18-17 文件管理界面

选择图 18-17 顶部的“树图管理”选项卡后来到一个新界面，如图 18-18 所示。

在“树图管理”界面中可以查看某个文件夹下的所有子文件，例如，图 18-19 显示了 system/app 目录下的文件。



图 18-18 “树图管理”界面

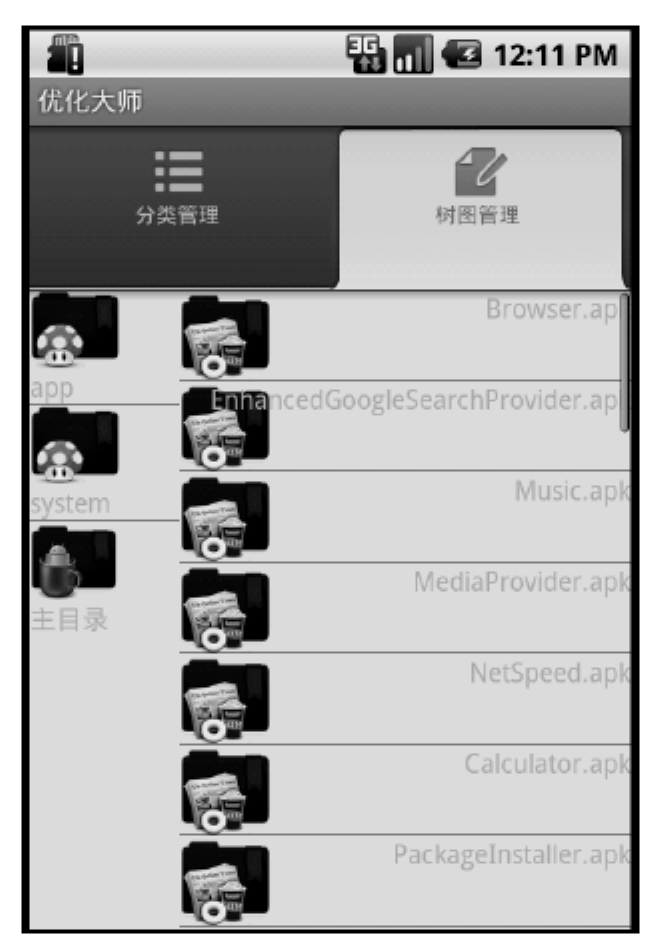


图 18-19 system/app 的子目录文件

